Universidade de Aveiro Departamento de Engenharia Mecânica
2016

Hugo Silva Ribeiro

Sistema de Monitorização de Energia

Universidade de Aveiro Departamento de Engenharia Mecânica
2016

Hugo Silva Ribeiro

Sistema de Monitorização de Energia

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação científica do Professor Doutor José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro

O Júri / The Jury

Presidente	Prof. Doutor Fernando José Neto da Silva Professor Auxiliar, Universidade de Aveiro
Orientador	Prof. Doutor José Paulo Oliveira Santos Professor Auxiliar, Universidade de Aveiro
Arguente	Prof. Doutor Joaquim José Borges Gouveia Professor Catedrático Convidado Aposentado, Universidade de Aveiro

Agradecimentos

Agradeço a toda a minha família pelo apoio que me foi dado ao longo de todo o meu percurso académico, em especial aos meus pais, o meu irmão, e os meus avós maternos. Com uma lembrança especial à minha avó que pode acompanhar em vida o desenvolvimento deste trabalho, mas infelizmente não pôde ver como acaba. Um agradecimento muito especial também à minha namorada que me apoiou imenso no desenvolvimento deste trabalho. Com apoio assim, todo o trabalho difícil se torna fácil.

Palavras chave energia, eficiência energética, SCADA, monitorização, autómato, contador de energia, protocolo, base de dados

Resumo

Este trabalho foi desenvolvido na Colep SA Protugal em Vale de Cambra ao abrigo do programa da Galp 202020, com o objetivo de encontrar e aplicar medidas com vista à melhoria da poupança energética.

Ao longo deste trabalho, foi projetado, desenvolvido e aplicado um sistema de monitorização de energia elétrica. O sistema de monitorização de energia proposto recorre a uma aplicação em Visual Basic .Net capaz de comunicar com autómatos programáveis e com os contadores de energia existentes na empresa, regista estes valores numa base de dados em Microsoft Access.

Da implementação de um sistema de monitorização de energia elétrica como o desenvolvido para esta empresa, pode ser atingida uma poupança de 2% a 8%. Se o sistema não é apenas de monitorização de energia elétrica, mas sim um sistema mais complexo e com possibilidade de controlo e outro tipo de análises, como é o caso dos sistemas SCADA, a poupança neste caso pode atingir os 10%. Para esta empresa, usando o valor intermédio de poupança de 5%, isto equivale a uma poupança de 982,8 MWh/ano, o que equivale a uma poupança de cerca de 461,9 tCO_2/ano. Quanto à poupança nos custos da energia elétrica é cerca de 86.546,7€/ano. Quanto ao Payback do sistema não pode ser calculado da forma tradicional, pois não é a implementação do sistema em si que permite a poupança da energia, mas sim da análise feita aos valores recolhidos à custa de um sistema destes que permite implementar medidas que permitam este valor de poupança.

Keywords: energy, energy efficiency, SCADA, monitoring, PLC, energy meter, protocol, data base

Abstract: This work was developed in Colep SA Portugal in Vale de Cambra under Galp 202020 program, in order to find and implement systems to improve energy savings in industry.

Throughout this work, it was designed, developed and applied a monitoring system of electric power. The energy monitoring system proposed uses an application in Visual Basic .NET able to communicate with PLCs and with existing energy meters in company, records these values in a Microsoft Access database.

Implementing a power monitoring system as developed for this company, a savings of 2% to 8% can be achieved. If the system is not only electric power monitoring, but a more complex system with the possibility of control and other types of analyzes, such as the SCADA systems, the saving in this case can reach 10%. For this company, using the intermediate value of 5% savings, this equates to a 982.8 MWh / year savings, which is equivalent to a saving of about 461.9 tCO2 / year. As for savings in the cost of electricity is about € 86,546.7 / year. As for the Payback system can not be calculated in the traditional way, it is not the implementation of the system itself that allows for energy savings, but the analysis of the amounts paid at the expense of such a system that can implement measures to this saving value.

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas

ADENE Agência para a Energia

ARCE Acordo de Racionalização dos Consumos de Energia

BMS Building Management Systems

CIE Consumidores Intensivos de Energia

CRC Cyclic-Redundant Checksum

DGEG Direção Geral de Energia e Geologia

ENE 2020 Estratégia Nacional para a Energia

FDL Field Bus Data Link

GEE gases com efeito estufa

HMI Interface Homem Máqquina, do inglês Human Machine Interface

IAS Industrial Automation Systems

JDBC Java Database Connectors

MES Manufacturing execution systems

ODBC Open Database Connectivity

OSI Open Systems Interconnect

PC Computador pessoal, do inglês Personal Computer

PLC Controlador Lógico Programável, do inglês Programmable Logic Controller

PNAC Programa Nacional para as Alterações Climáticas

PNAEE Plano Nacional de Ação em Eficiência Energética

PNAER Plano Nacional de Ação para as Energias Renováveis

PREn Planos de Racionalização dos Consumos de Energia

REP Relatórios de Execução e Progresso

RTU Unisdade Terminal Remota, do inglês Remote Terminal Unit

SCADA Sistemas de Supervisão e Aquisição de Dados, do inglês Supervisory Control and Data Acquisition

SGCIE Sistema de Gestão dos Consumos Intensivos de Energia

SQL Linguagem de Consulta Estruturada, do inglês Structured Query Language

TEP tonelada equivalente de petróleo

UV Ultravioletas

VAB Valor Acrescentado Bruto

XML eXtensible Markup Language

1 Introdução

1 Introdução

1.1 Objetivos

O objetivo deste trabalho foi desenvolver um sistema de monitorização de energia de baixo custo e o mais versátil possível aplicável à empresa Colep Portugal SA.

1.2 Enquadramento

Com a crescente preocupação tanto em termos económicos como ambientais, é perentório olhar para a utilização que se faz da energia por forma a fazer uma gestão mais cuidada. Os vários setores da sociedade estão empenhados em reduzir o consumo de energia por vários motivos. Por um lado, a indústria tem como objetivo reduzir os custos da produção, por outro lado o objetivo dos governos é diminuir a quantidade de energia consumida pelas indústrias e particulares para que os seus países não estejam tão dependentes de energias não renováveis, de que muitas vezes não dispõem e por isso têm de as importar. Para combater e tentar diminuir esta dependência usam muitas vezes a legislação como ferramenta. Existe também nos últimos anos uma grande preocupação ambiental por parte da sociedade em geral. A somar a todos estes fatores existe ainda também o problema da escassez de energia que faz com que o preço aumente dia após dia.

A Europa enfrenta desafios sem igual na sua história resultantes do aumento da dependência da importação de energia, escassez de recursos energéticos e da necessidade de conter as alterações climáticas. A União Europeia, através da diretiva 2012/27/EU, define a eficiência energética como sendo uma ferramenta fundamental para ultrapassar estes desafios. Um outro objetivo que passa pela eficiência energética e está contido no plano europeu “Estratégia Europa 2020” é o de dissociar o crescimento económico do consumo de energia.

No caso português e no seguimento da diretiva 2012/27/EU existe um plano a nível nacional para que o país não esteja tão dependente em termos energéticos. Este plano tem o nome de Estratégia Nacional para a Energia (ENE 2020) e assenta sobre cinco eixos [1].

- Eixo 1 – Agenda para a competitividade, o crescimento e a independência energética e financeira.
- Eixo 2 – Aposta nas energias renováveis.
- Eixo 3 – Promoção da eficiência energética.
- Eixo 4 – Garantia da segurança de abastecimento.
- Eixo 5 – Sustentabilidade económica e ambiental.

Nesta Estratégia Nacional para a Energia está inserido o PNAEE - Plano Nacional de Ação para a Eficiência Energética que tem como meta uma poupança de 8,2% até 2016 [2]. Os contributos na redução dos consumos energéticos estão distribuídos pelos setores de atividade seguintes:

- Transportes
- Residencial e Serviços
- Indústria
- Estado
- Comportamentos e Agricultura

Em 2012 estes setores tinham a distribuição representada na Figura 1.1. Como é possível ver a indústria tem um peso de 32,5% na energia consumida por setor em Portugal.

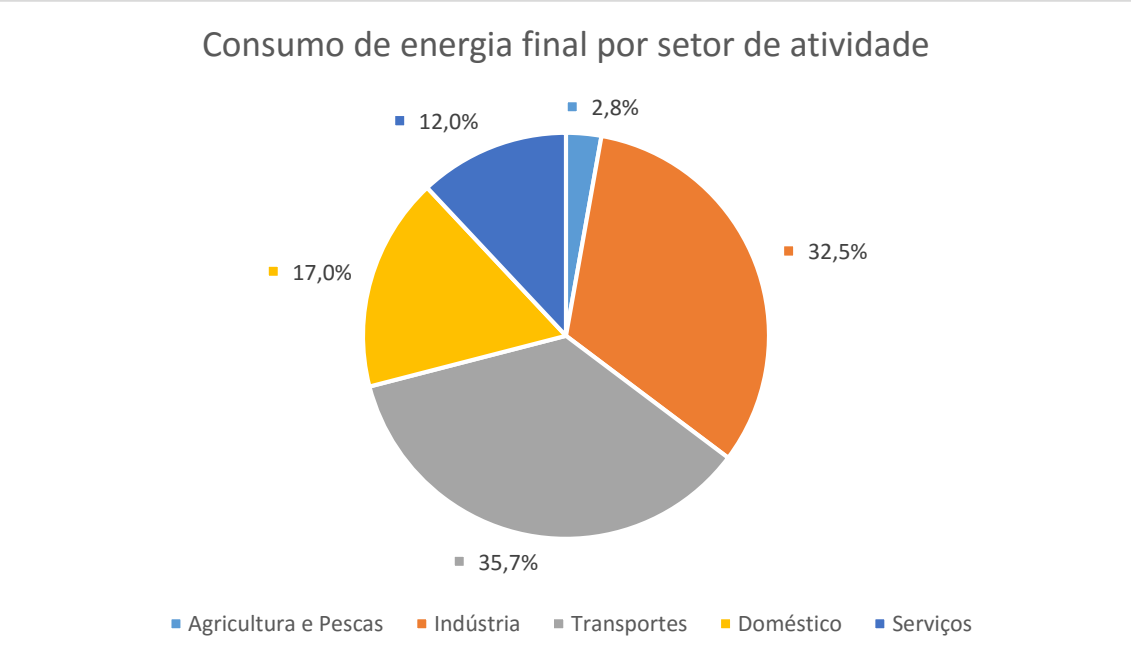


Figura 1.1 – Consumo de Energia final por setor de atividade [3]

No caso particular da indústria portuguesa e no âmbito da Estratégia Nacional para a Energia o Decreto-Lei n.º 71/2008, de 15 de Abril, faz o regulamento do SGCIE [4] – Sistema de Gestão dos Consumos Intensivos de Energia, aplicado às instalações consumidoras intensivas de energia (CIE) com consumos superiores a 500 tep/ano [5]. O SGCIE determina que no mínimo haja uma melhoria de 6% da Intensidade Energética e do Consumo Específico de Energia, em seis anos, quando se trate de instalações com consumo intensivo de energia igual ou superior a 1000 tep/ano, ou melhoria de 4% em oito anos para as restantes instalações. Quanto à intensidade carbónica, terá que no mínimo se manter relativamente aos valores históricos. As empresas que estejam abrangidas por um Acordo de Racionalização dos Consumos de Energia (ARCE) têm direito aos seguintes incentivos[6]:

- No caso de instalações com consumos inferiores a 1000 tep/ano - Ressarcimento de 50% do custo das auditorias energéticas obrigatórias, até ao limite de € 750
- Ressarcimento de 25% dos investimentos realizados em equipamentos e sistemas de gestão e monitorização dos consumos de energia até ao limite de € 10 000

No caso das instalações que consumam apenas gás natural como combustível e/ou energias renováveis, os limites previstos anteriormente são majorados em 25% no caso das renováveis e 15% no caso do gás natural.

Tendo em conta o SGCIE, pode ser feito um enquadramento da empresa segundo o consumo energético anual, e consequentemente saber as metas de poupança obrigatórias segundo a legislação em vigor assim como os ressarcimentos possíveis de obter nos gastos com sistemas de monitorização dos consumos de energia. No capítulo seguinte irá ser feita a caracterização da empresa em estudo quanto ao seu processo produtivo e aos consumos de energia associados.

1.3 Caso de Estudo/Contributo cientifico

Com o sistema de monitorização de energia desenvolvido neste trabalho, é possível fazer uma gestão da energia recorrendo-se dos valores que são obtidos. Este tipo de sistemas são fundamentais para a gestão da energia na industria mas são muito dispendiosos. Neste trabalho a solução apresentada é bastante barata e consegue realizar bem a tarefa a que se propõe. Utilizando equipamento standard e *software* desenvolvido numa linguagem de programação acessível, torna-se numa alternativa fiável às soluções mais comuns que normalmente recorrem a *software SCADA* que têm licenças dispendiosas.

1.4 Organização do Documento

Este documento encontra-se dividido em 6 capítulos:

- O capítulo 1 introduz a temática e os objetivos do trabalho, bem como os problemas a resolver;
- No capítulo 2 a empresa onde se desenvolveu o trabalho, é caracterizada em termos energéticos e explicado o seu processo produtivo;
- O capítulo 3 explica a legislação europeia e portuguesa sobre eficiência energética, e são apresentados os conceitos teóricos que foram úteis ao desenvolvimento deste trabalho;
- No capítulo 4 é explicada a solução apresentada para resolver o problema, e de que maneira foi desenvolvida na prática
- No capítulo 5 são apresentados os resultados de investimento e consequente poupança energética
- No capítulo 6 são tiradas as conclusões deste trabalho

2 Caracterização da Empresa em Estudo

2 Caracterização da Empresa em Estudo

2.1 Descrição da Empresa e História

A Colep Portugal, S.A. situada em Vila Chã – Vale de Cambra, fundada em 1965 deu início à sua atividade com o fabrico de embalagens metálicas de decoração. Em 1967 foi alargada a atividade de fabrico às embalagens metálicas industriais. No ano de 1975 é dado um passo importante na expansão da empresa com o início da atividade de *Contract Manufactering*, que consiste na formulação e fabrico, enchimento e embalamento de vários produtos. Atualmente a Colep dedica-se ao fabrico de embalagens para aerossóis, alimentares e indústria e também se dedica à produção de embalagens plásticas.

Atualmente a Colep pertence ao Grupo RAR e é líder mundial na indústria de bens de consumo e no seu embalamento. As várias fábricas da Colep empregam cerca de 3800 pessoas em países como Portugal, Brasil, Espanha, Emirados Árabes Unidos e Reino Unido.

No entanto a empresa tem representações em muitos outros países como mostra a Figura 2.1.

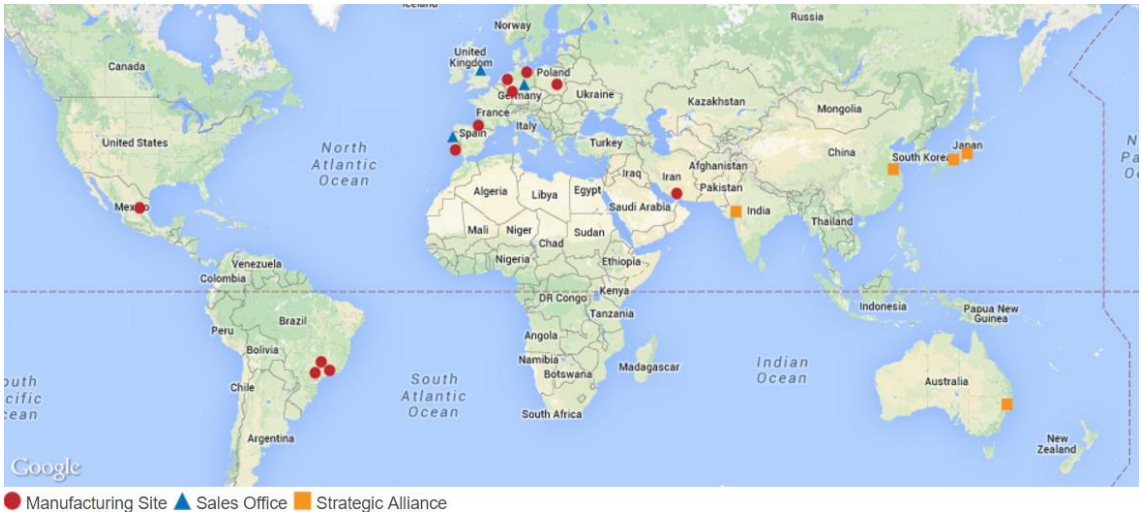


Figura 2.1 – Grupo Colep pelo mundo [1]

2.2 Processo Produtivo

A fábrica da Colep situada em Vale de Cambra que foi o local onde foi realizado o estágio, desenvolve a sua atividade no fabrico de embalagens metálicas e plásticas, no enchimento de produtos em regime de *contract filling* e no embalamento de produtos em regime de subcontratação.

As várias secções produtivas dividem-se em cinco áreas que estão representadas abaixo na Figura 2.2.

1. Litografia
2. Embalagens Metálicas
3. Embalagens Plásticas
4. Enchimento
5. *Co-packing*

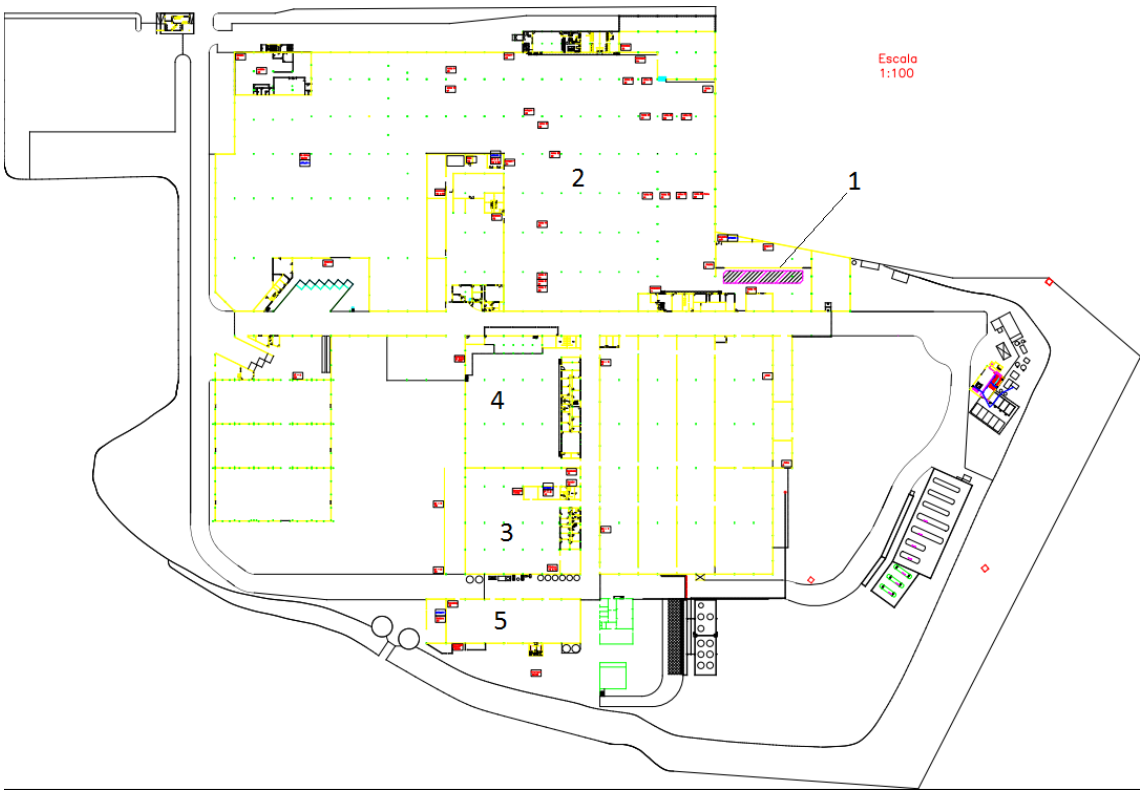


Figura 2.2 – Planta da fábrica de Vale de Cambra

2.2.1 Litografia

O regime de funcionamento desta secção é de três turnos, 24 horas/dia, das 6 horas de 2ª-feira às 6 horas de sábado. O processo produtivo desta secção desenvolve-se em dois sectores principais: *Littell* e Litografia. Este processo começa com a receção das bobinas de chapa metálica (folha de flandres) e o seu no respetivo armazém a partir do qual é feita a alimentação da chapa à máquina *Littell*. Esta máquina desenrola as bobinas, corta a chapa e agrupa-a em lotes. Os lotes são embalados e armazenados para serem enviados para o sector seguinte. A litografia das chapas é feita em dois grupos de linhas de impressão e secagem. Há um grupo cuja secagem das chapas após impressão é feita através de lâmpadas Ultravioletas (UV) e há um outro grupo que têm associados fornos a gás natural para secagem das chapas após impressão.

2.2.2 Embalagens Metálicas

O regime de funcionamento desta secção é de quatro turnos, 24 horas/dia, das 6 horas de 2ª-feira às 6 horas de sábado, com a exceção da linha de aerossóis que funciona em regime contínuo. Na secção de corte, as chapas vão ser cortadas em diversos formatos conforme o tipo de embalagem a que se destinam. Com as chapas cortadas são produzidas embalagens que passam por dois processos distintos: a estampagem e a montagem das embalagens.

Na estampagem são utilizadas prensas para conformar as diversas peças que constituem uma embalagem. Em algumas linhas da estampagem existem fornos de gás natural para secagem das borrachas de vedação colocadas nas embalagens.

Na montagem todas as peças que irão dar origem à embalagem final são reunidas e montadas. Certas linhas da montagem possuem fornos de gás natural responsáveis pela secagem do verniz colocado na zona de soldadura. Com as embalagens montadas é feito um teste de estanquidade através da pressurização das mesmas com ar comprimido.

2.2.3 Embalagens Plásticas

O regime de funcionamento das linhas de injeção é de três turnos, 24 horas/dia, das 6 horas de 2ª-feira às 6 horas de sábado, enquanto o regime de funcionamento das linhas de insuflação é contínuo. O processo produtivo desta secção desenvolve-se em três sectores: insuflação, injeção e serigrafia.

Para o fabrico de embalagens pelo método de insuflação são utilizadas dezasseis máquinas onde é feito o aquecimento das matérias-primas através de resistências elétricas, a extrusão, a insuflação com ar comprimido para o interior de um molde e, por último, a rebarbagem.

Para fabrico de embalagens por injeção existem onze máquinas onde é feito o aquecimento das matérias-primas por intermédio de resistências elétricas, a injeção num molde e, finalmente, o arrefecimento da embalagem.

Há ainda uma máquina de serigrafia e uma máquina de rotulagem por onde passam algumas embalagens.

2.2.4 Enchimento

O regime de funcionamento desta secção é de três turnos, 24 horas/dia, das 6 horas de 2ª-feira às 6 horas de sábado. O processo produtivo desta secção desenvolve-se em três sectores: formulação de produtos, enchimento de aerossóis e enchimento de líquidos.

A maior parte das embalagens utilizadas para enchimento nesta secção são provenientes das secções de Embalagens Metálicas e de Embalagens Plásticas da fábrica, havendo algumas que são provenientes de empresas externas.

A formulação dos produtos é feita em depósitos existentes numa sala independente da sala de enchimento.

Depois de feita a formulação, os produtos são armazenados em depósitos a partir dos quais é feita a alimentação das linhas de enchimentos.

2.2.5 Co-Packing

O regime de funcionamento desta secção é de um turno, 8 horas/dia, das 6 horas de às 14 horas, de 2ª-feira a 6ª-feira. No período de referência da auditoria o regime de funcionamento desta secção era de dois turnos. Neste sector são embalados artigos em regime de subcontratação. Normalmente, são recebidos do exterior os artigos em pequenas séries de peças soltas para serem montados e embalados nas linhas existentes, resultando os produtos finais desta secção.

O diagrama do processo produtivo encontra-se na Figura 2.3.

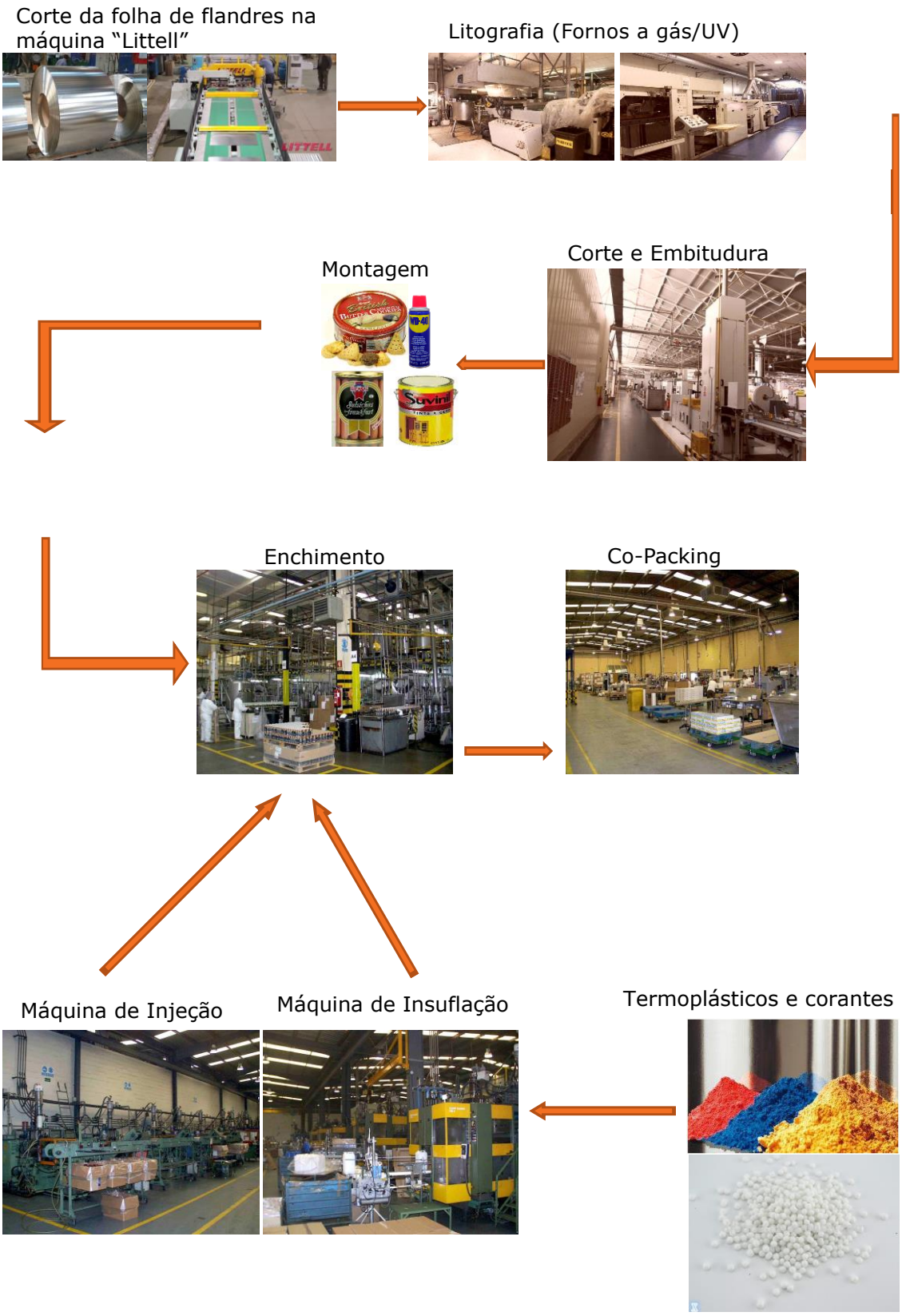


Figura 2.3 – Diagrama do processo produtivo

2.3 Consumos e Custos da Energia

Nesta empresa existem 6 formas de energia consumidas: Vapor, Água Quente, Gás Natural, Gasóleo, Gasolina, Energia Elétrica. Os consumos e custos anuais da energia estão expostos na Tabela 2.1. A energia representada em GJ encontra-se na forma final de consumo, enquanto que a coluna em que a energia está representada em tep, os valores encontram-se em valores de energia primária.

Tabela 2.1 - Consumos e custos da Energia

Ano 2011	Quantidade	Energia		Energia		Custo da Energia		Gases Efeito Estufa	
		GJ	%	tep	%	€	%	tCO_{2e}	%
Vapor	4,172.8 t	9,819	6.2%	261	4.1%	178,280	6.4%	710	4.7%
Água Quente	666.1 MWh	2,398	1.5%	64	1%	39,063	1.4%	173	1.2%
Gás Natural	1,579.8 t	71,248	45%	1,701	26.8%	698,824	25.2%	4,566	30.4%
Gasóleo	93.5 t	4,049	2.6%	97	1.5%	124,842	4.5%	300	2.0%
Gasolina	2.8 t	125	0.1%	3	0%	4,633	0.2%	9	0.1%
Energia Elétrica	19,656.7 MWh	70,764	44.7%	4,226	66.5%	1,730,954	62.3%	9,238	61.6%
Total		158,403	100%	6,352	100%	2,776,596	100%	14,996	100%

Para poder averiguar melhor a distribuição dos consumos de energia, a informação da Tabela 2.1 foi traduzida num gráfico como mostra a Figura 2.4:

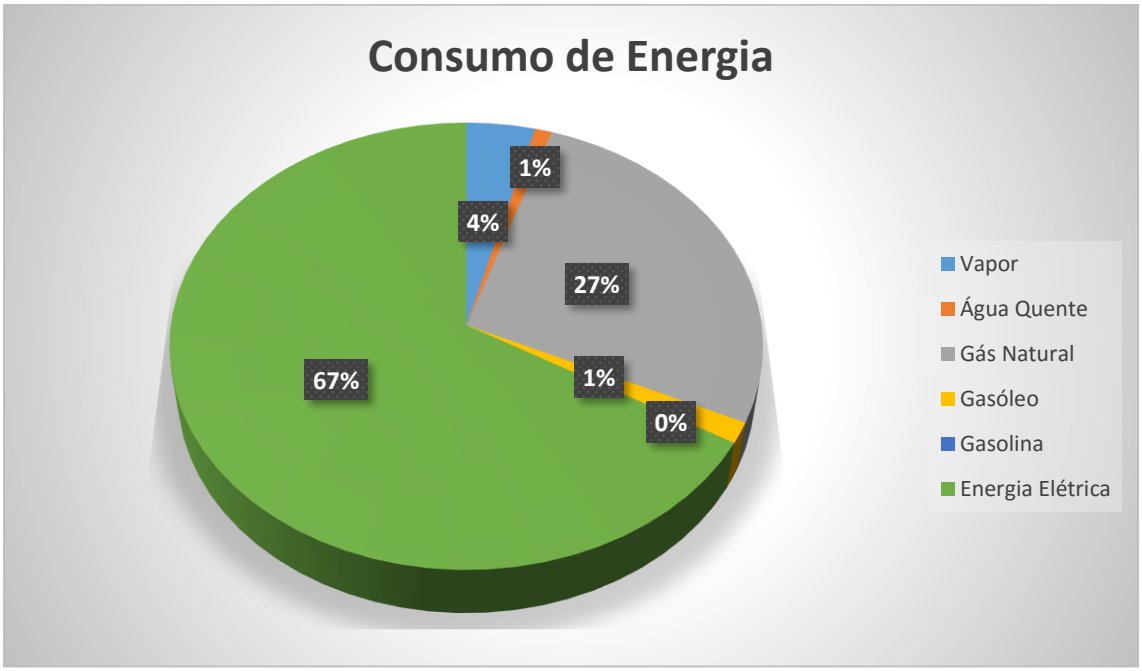


Figura 2.4 – Consumo de Energia em tep

Foi fácil verificar através da Figura 2.4 que a forma de energia com mais peso tanto na fatura como em termos de consumo absoluto em tep foi a energia elétrica. Foi assim então sobre este

tipo de energia que este trabalho incidiu. No entanto é possível verificar que o consumo de gás natural também é elevado. Neste trabalho, o sistema de monitorização de energia vai incidir apenas na energia elétrica, mas este tipo de solução pode também ser aplicado ao gás natural. Outras soluções de poupança para o gás natural poderiam passar por remodelar as linhas de secagem das chapas vindas da litografia, que são na sua maioria a gás, e substituí-las por linhas de secagem UV. Note-se que uma poupança de gás natural equivalente a uma poupança na energia elétrica, tendo como base de comparação em tep, a diminuição de emissões de GEE é maior no caso do gás natural. Além disso, um sistema de monitorização de energia aplicado aos consumos de gás natural tem poupanças estimadas entre 5 e 15%, enquanto que no caso deste sistema apenas se aplicar à energia elétrica a poupança prevista situa-se entre os 2 e 8% [7].

3

Estado da Arte

3 Estado da Arte

O crescimento da população mundial e o aumento consequente dos níveis de consumo nas últimas décadas têm colocado o problema da sustentabilidade dos recursos energéticos que permitem o desenvolvimento das civilizações. Outro problema devido a este excesso de consumo é a questão ambiental. Nos últimos anos este problema da sustentabilidade quer pelo lado da gestão de recursos quer pelo lado ambiental tem sido olhado de uma maneira mais séria pelas grandes potências mundiais. Assim têm sido frequentes as cimeiras e os acordos assinados com promessas de melhorias na gestão destes recursos e da proteção ambiental. O problema, no entanto, é complicado pois se por um lado há um excesso de consumo e utilização destes recursos, por outro lado o desenvolvimento das civilizações depende diretamente deste consumo. Ora esta equação só pode ser alterada por duas variáveis, ou pela redução dos consumos energéticos, ou pelo aumento da eficiência da sua utilização. Uma outra maneira de contornar a escassez dos recursos energéticos é a utilização de recursos energéticos renováveis como o caso da energia solar, eólica, marés, etc.

Neste trabalho é apresentada uma solução para aumentar a eficiência da energia utilizada. Nas secções seguintes irá ser abordada a legislação a nível internacional e a legislação nacional sobre esta temática da energia.

3.1 Análise da legislação europeia

A diretiva 2012/27/EU do parlamento e do concelho europeus de 25 de outubro de 2012, alterou as diretivas 2009/125/CE e 2010/30/EU e revogou as diretivas 2004/8/CE e 2006/32/CE, publicadas anteriormente. Esta diretiva mostra a preocupação europeia com a dependência de importações de energia conjugada com a cada vez maior escassez destes recursos provocando a escalada de preços que tem contribuído para acentuar a crise que se vive na europa. Assim esta dependência do exterior e o perigo das alterações climáticas devidas em grande parte às emissões de gases com efeito de estufa (GEE) para a atmosfera, fazem com que a união europeia reconheça a eficiência energética como o instrumento mais capaz para combater estes problemas. Uma economia mais eficiente do ponto de vista energético é também olhada por parte da união europeia como possível impulsionadora da melhoria da competitividade da indústria o que ajudará ao crescimento económico e criação de postos de trabalho[8].

Do conselho europeu de 8 e 9 de março de 2007 foram colocados como objetivos a redução dos consumos de energia primária até 2020 em cerca de 20% na UE tendo como base de cálculo as projeções de consumo de energia. Em 2007 foram feitas projeções do consumo de energia primária em 2020 de 1 842 Mtep. Uma redução deste número em 20% até ao ano de 2020 corresponde a um consumo de energia primária em cerca de 1 474 Mtep [8].

No caso português, o país determinou uma Estratégia Nacional para a Energia para 2020 (ENE 2020) com o objetivo "liderar a revolução energética" colocando como meta "assegurar a posição de Portugal entre os cinco líderes europeus ao nível dos objetivos em matéria de energias renováveis em 2020 e afirmar Portugal na liderança global na fileira industrial das energias renováveis, de forte capacidade exportadora". Da ENE 2020 constam metas para reduzir a dependência energética do país em relação ao estrangeiro como: reduzir a dependência energética do país face ao exterior para 74 % em 2020, produzindo, nesta data, a partir de recursos endógenos, o equivalente a 60 milhões de barris anuais de petróleo, com vista

à progressiva independência do País face aos combustíveis fósseis; garantir o cumprimento dos compromissos assumidos por Portugal no contexto das políticas europeias de combate às alterações climáticas, permitindo que em 2020 60 % da eletricidade produzida e 31 % do consumo de energia final tenham origem em fontes renováveis e uma redução do 20 % do consumo de energia final nos termos do Pacote Energia-Clima 20-20-20; reduzir em 25% o saldo importador energético com a energia produzida a partir de fontes endógenas gerando uma redução de importações de 2000 milhões de euros; criar riqueza e consolidar um cluster energético no sector das energias renováveis em Portugal, assegurando em 2020 um valor acrescentado bruto de 3800 milhões de euros e criando mais 100 000 postos de trabalho a acrescer aos 35 000 já existentes no sector e que serão consolidados. Dos 135 000 postos de trabalho do sector, 45 000 serão diretos e 90 000 indiretos. O impacto no PIB passará de 0,8 % para 1,7 % até 2020; desenvolver um cluster industrial associado à promoção da eficiência energética assegurando a criação de 21 000 postos de trabalho anuais, gerando um investimento previsível de 13 000 milhões de euros até 2020 e proporcionando exportações equivalentes a 400 milhões de euros; promover o desenvolvimento sustentável criando condições para o cumprimento das metas de redução de emissões assumidas por Portugal no quadro europeu. [9].

A ENE2020 assenta nos seguintes 5 eixos principais: agenda para a conetividade, o crescimento e a independência energética e financeira; aposta nas energias renováveis; promoção da eficiência energética, garantia da segurança de abastecimento; sustentabilidade ambiental. Assim o objetivo desta agenda é funcionar como um fator de crescimento da economia, promoção da concorrência dos mercados de energia, criação de valor e de emprego qualificado em setores com elevada incorporação tecnológica [1].

3.2 Plano Nacional de Ação e Eficiência Energética (PNAEE)

Tendo em conta as metas europeias “20-20-20” que têm como objetivo alcançar no ano de 2020 uma redução de 20% nas emissões de gases com efeito de estufa tendo como valor de referência os valores do ano de 1990, que 20% do consumo de energia final bruto seja proveniente de energias renováveis, e também uma redução de 20% do consumo de energia primária tendo como valor de referência os valores projetados para 2020, Portugal traçou o objetivo para o ano de 2020 reduzir o consumo de energia primária em cerca de 25% e um objetivo específico para a função pública de 30%, usando para isso o aumento da eficiência energética [10].

Para auxiliar estes objetivos existem planos de ação para o aumento da eficiência energética em Portugal. Em baixo estão listados os principais:

- Plano de Ação para a Eficiência Energética (PNAEE)
- Plano Nacional de Ação para as Energias Renováveis (PNAER)
- Programa Nacional para as Alterações Climáticas (PNAC)

O programa inicial do PNAEE com data de 2008, estabelecia uma meta de redução de 10% do consumo de energia final até o final 2015. No entanto posteriormente as metas foram alteradas para uma redução de 20% no consumo de energia final até 2020 através da ENE2020. O PNAEE de 2008 previa um pacote de 50 medidas, distribuídas em 12 programas e associados a 4 setores [10].

Posteriormente o PNAEE sofreu uma modificação para a versão PNAEE 2016 que colocou como meta uma poupança induzida de 1501 ktep (em energia final), correspondente a uma redução do consumo energético de aproximadamente 8,2% relativamente à média do consumo

verificada no período entre 2001 e 2005, o que se aproxima da meta indicativa definida pela União Europeia de 9% de poupança de energia até 2016 [2].

O pacote de medidas constituintes do PNAEE destinados ao setor da indústria, estão contidos no programa SGCIE (Sistema de Gestão de Consumos Intensivos de Energia) que vai ser explicado na secção seguinte.

3.3 Sistema de Gestão de Consumos Intensivos de Energia

O SGCIE destina-se a todas as instalações consumidoras intensivas de energia (CIE) que no ano civil antecedente tenham atingido um consumo energético superior a 500 tep (tonelada equivalente de petróleo). São exceção as instalações de cogeração juridicamente autónomas dos respetivos consumidores de energia [11].

Este regulamento prevê que as instalações consumidoras intensivas de energia realizem regularmente auditorias energéticas que como resultado têm Planos de Racionalização dos Consumos de Energia (PREn). "Os PREn, quando aprovados, constituem Acordos de Racionalização dos Consumos de Energia (ARCE) celebrados com a DGEG, associando ao seu cumprimento a obtenção de incentivos pelos Operadores dessas instalações" [11].

Desta forma, qualquer instalação CIE que adira ao SGCIE, compromete-se a realizar uma série de tarefas descritas na Figura 3.1

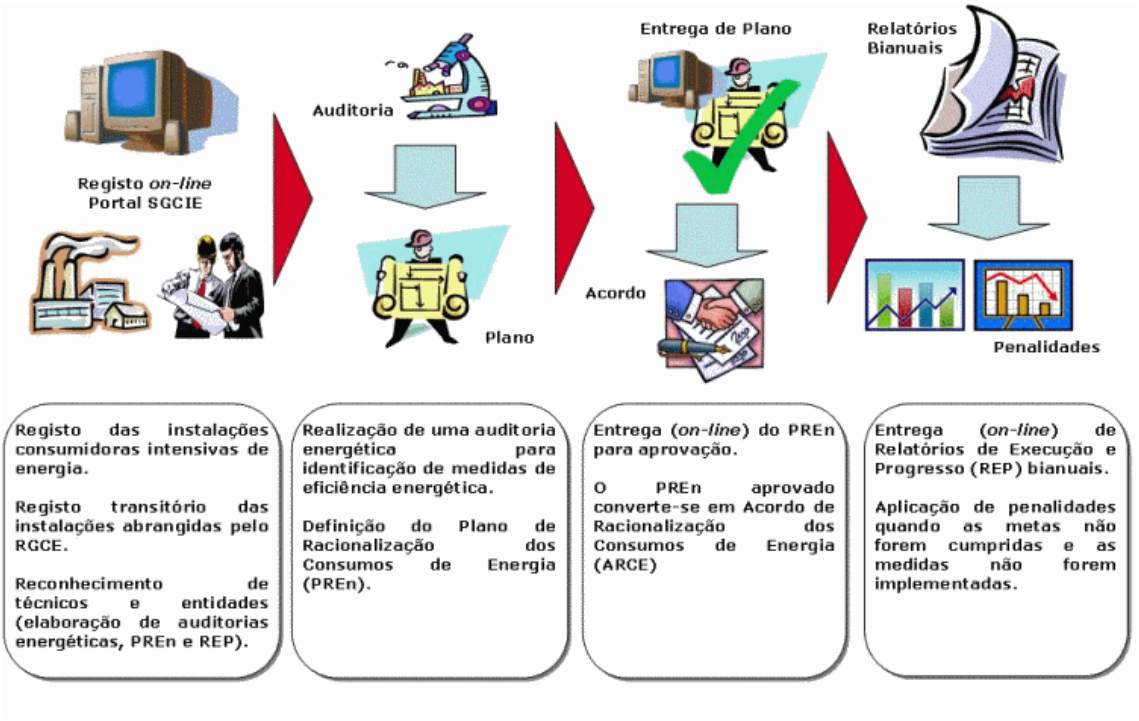


Figura 3.1 – Procedimento do SGCIE [11]

Essa sequência de tarefas é destinada à unidade CIE, e são as seguintes:

- Efetuar o registo das instalações no portal do SGCIE
- Efetuar auditorias energéticas no sentido de identificar medidas que aumentem a eficiência energética da empresa e consequentemente reduzam os custos com a energia
- Elaborar um PREn resultante da análise dos resultados da auditoria e apresenta-lo à ADENE (Agência para a Energia)

- Executar o plano por forma a cumprir o Acordo de Racionalização dos Consumos de Energia (ARCE), que não é mais que o PREn aprovado
- Efetuar Relatórios de Execução e Progresso (REP) bianuais
- No caso de as metas não serem cumpridas, serão aplicadas penalidades

3.3.1 Escalões do SGCIE

Existem dois escalões para a implementação do SGCIE para os quais se enquadram as unidades CIE. A divisão é feita com base no consumo energético no ano imediatamente anterior à adesão ao programa. Esses escalões são os seguintes:

- Entre 500 tep e 1000 tep
- Superior a 1000 tep

As instalações com consumo energético inferior a 500 tep/ano apesar de não estarem obrigadas a aderir ao SGCIE, podem fazê-lo de forma voluntária e celebrar Acordos de Racionalização do Consumo de Energia com a DGEG [12].

Quando elaborado o PREn, com base nas auditorias energéticas, as medidas acordadas têm de ser implementadas nos primeiros 3 anos. O período de retorno dessas medidas para as instalações CIE é de 3 anos para as instalações com consumos entre 500 e 1000 tep/ano, e de 5 anos para as instalações com consumo superior a 1000 tep/ano [5].

O PREn tem também de estabelecer metas relativas a indiciadores específicos. Esses indicadores são:

- Intensidade energética
- Intensidade carbónica
- Consumo específico de energia

A Figura 3.2 explica o significado de cada um destes indicadores.

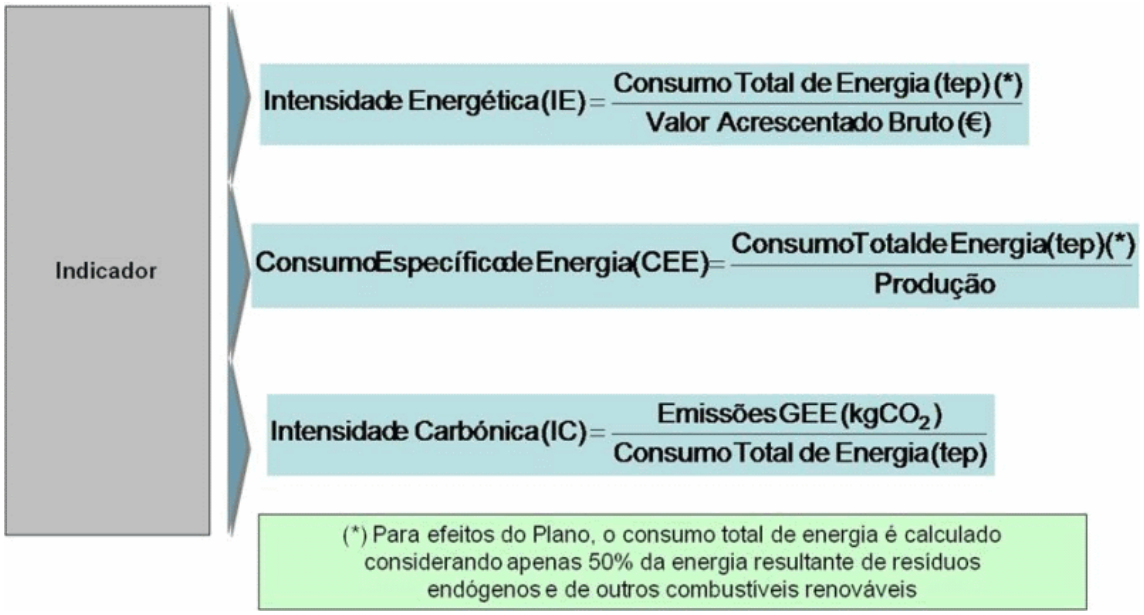


Figura 3.2 – Indicadores de consumo energético [5]

A Intensidade Energética é o resultado do quociente entre o consumo total de energia (considerando apenas 50% da energia resultante de resíduos endógenos e de outros combustíveis renováveis) e o Valor Acrescentado Bruto (VAB) das atividades empresariais diretamente ligadas a essas instalações industriais [5].

Intensidade Carbónica é o resultado do quociente entre o valor das emissões de GEE resultantes da utilização das várias formas de energia no processo produtivo e o respetivo consumo total de energia [5].

Consumo Específico de Energia é o resultado do quociente entre o consumo total de energia (considerando apenas 50% da energia resultante de resíduos endógenos e de outros combustíveis renováveis) e o volume de produção [5].

Quanto às metas de melhoria destes indicadores, são diferentes dependendo do escalão em que a instalação CIE se insere.

- Para instalações com consumos superiores a 1000 tep/ano, a intensidade energética e o consumo específico de energia, têm de ter uma melhoria de 6% em 6 anos.
- Para instalações com consumos superiores a 500 tep/ano, a intensidade energética e o consumo específico de energia, têm de ter uma melhoria de 4% em 8 anos.

3.3.2 Incentivos e penalizações

No caso de uma instalação não conseguir cumprir as metas acordadas, ou não implemente as medidas também acordadas no ARCE, e nos casos em que no ano seguinte ao relatório final de execução a instalação não recupere os desvios, implica [13]:

A Figura 3.3 mostra de forma simples o sistema de penalização.

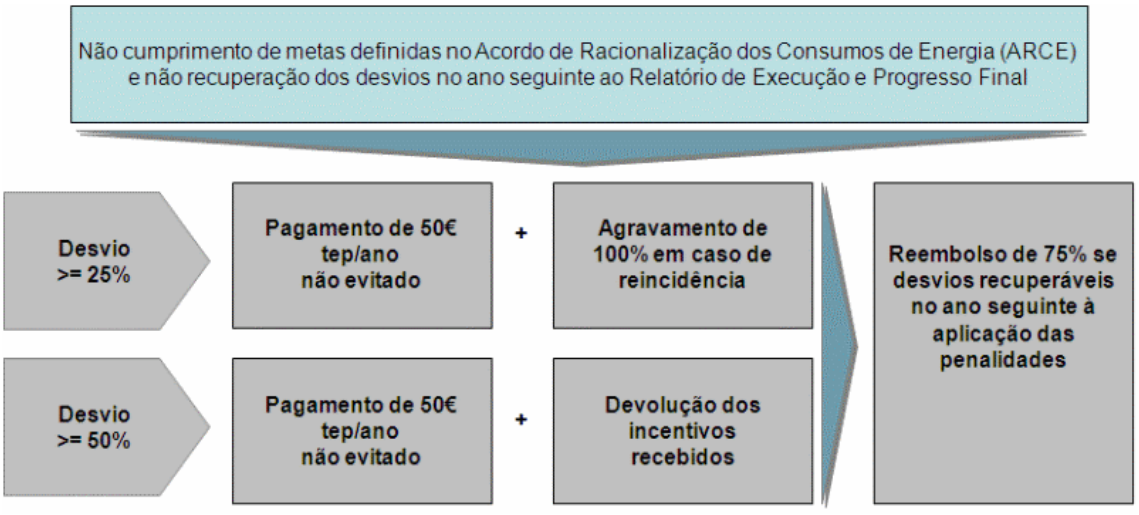


Figura 3.3 – Sistema de penalizações [13]

- No final do período a que o ARCE foi acordado, se o desvio for igual ou superior a 25%, o terá de pagar € 50/tep/ano que não foi evitado como previsto. No caso de reincidência o valor é agravado em 100% [13];
- No final do período a que o ARCE foi acordado, se o desvio for igual ou superior a 50%, para além de pagar dos valores do ponto anterior a ser pagos, terão de ser pagos os possíveis apoios que foram recebidos e de possíveis benefícios [13].

Se no ano seguinte à penalização, o operador conseguir recuperar, pode reembolsar 75% das penalizações, mediante despacho do Diretor-geral da Direção Geral de Energia e Geologia (DGEG) [13].

Contudo, também existe benefícios para o cumprimento destas metas acordadas no ARCE. Um operador que tenha efetuado o ARCE beneficia das seguintes ajudas:

- No caso de instalações CIE com consumos abaixo dos 1000 tep/ano, estas beneficiam do ressarcimento de 50% do custo das auditorias energéticas obrigatórias, até ao limite de € 750, desde que na altura da execução do REP se verifique a aplicação de pelo menos 50% das medidas.
- Para qualquer instalação CIE é possível obter o ressarcimento de 25% do valor de investimentos realizados em equipamentos de monitorização e gestão dos consumos de energia até ao limite de € 10 000. Se a instalação em causa apenas consumir gás natural como energia e/ou energias renováveis, este ressarcimento é majorado em 25% nos caso das renováveis e 15% no caso do gás natural [6].

3.4 Sistemas SCADA

A monitorização de energia na industria normalmente está associada à monitorização dos processos produtivos. Esta monitorização dos processos é feita na maioria dos casos por Software SCADA (*Supervisory Control and Data Acquisition*). Um *software* SCADA é um *software* usado para controlo, monitorização e análise de um processo produtivo, variáveis de um edifício ou outro tipo de processo. Estes sistemas são usados praticamente em todos os processos produtivos ou industriais no munto. Alguns exemplos são os sistemas de água e esgotos, indústrias petroquímicas, indústria alimentar, etc. Estes sistemas utilizam telemetria, para fazer a recolha de dados dos dispositivos situados no chão de fábrica. Estes dispositivos muitas vezes encontram-se dispersos e medem variáveis associadas ao processo a monitorizar. O sistema SCADA reúne esses dados em centrais ondes estes são processados e guardados. Nestas centrais os operadores podem acompanhar o processo através de informação tratada em forma de gráficos, tabelas, etc. Muitas vezes através destas centrais os operadores podem acionar algum tipo de controlo sobre o processo tomando decisões decorrentes da análise que fazem dessa informação do processo que é fornecida em tempo real [14].

Um sistema SCADA começa pela comunicação, em tempo real, com os aparelhos no terreno que medem variáveis do processo. Tipicamente estes aparelhos tratam-se de PLC (Programmable Logic Controller) ou RTU (Remote Terminal Unit). O sistema reúne a informação destes aparelhos em tempo real para o software SCADA onde depois é mostrada através de interface gráfica aos operadores responsáveis do processo. Estes operadores podem assim responder às informações e alarmes mostrados pela interface, intervindo de imediato no processo e ajustando definições. Este tipo de software normalmente também produz históricos de informação através de bases de dados associadas, permitindo a criação de gráficos, ou a geração de relatórios. Desta maneira os operadores para além de acompanharem a evolução do processo em tempo real, podem avaliar esta informação acumulada ao longo do tempo permitindo assim fazer uma previsão do comportamento do sistema no futuro.[14]

3.4.1 Exemplos de *software SCADA/HMI*:

3.4.1.1 Ignition

O *Inginition* é um *software SCADA/HMI (Human Machine Interface)* da empresa *Inductive Automation*. Este *software* destaca-se por usar a tecnologia *Java Web Start* da *Oracle*, que faz com que este software trabalhe em qualquer máquina independentemente do sistema operativo.

Devido à sua arquitetura de servidor central, é preciso apenas instalar o *software* no servidor. Quanto aos clientes não precisam de instalar o *software*. Quando é necessário fazer um *update* no *software*, apenas é preciso instalar no servidor central. Uma vez atualizado o servidor, todos os clientes estão atualizados automaticamente. Isto é válido também para as modificações ao projeto que podem ser realizadas e publicadas no mesmo instante para todos os clientes.

Cada licença tem um número ilimitado de clientes permitidos. Estes clientes podem fazer login através da rede da fábrica ou através da internet.

A informação é guardada numa base de dados *standard SQL* (Structured Query Language). Isto significa que o utilizador pode retirar a informação a qualquer momento sem se preocupar com o tipo de armazenamento da base de dados. Assim é possível reunir toda a informação num só local e ganhar uma nova perspetiva da informação reunida. Este software é *built-in Java Database Connectors* (JDBC) para *MySQL, Microsoft SQL Server*, *Oracle* e mais. Não há limites para o número de conexões a bases de dados.

Inginition é escrito em Java. O que diferencia o Java das outras linguagens é a filosofia "escrito uma vez, corre em qualquer lado". Esta filosofia presente no *Inginition* permite correr um projeto independentemente da plataforma.

Manufacturing execution systems (MES) são sistemas usados na indústria que servem para fazer a localização/seguimento e documentar a transformação de matéria-prima em produto acabado. Este tipo de *software* permite fazer a ligação entre o chão de fábrica e a gestão da produção. A *Inductive Automation* oferece uma série de módulos *MES* para adicionar ao *Inginition*. Os módulos disponíveis são os seguintes:

- **Scheduling** – Planeia e monitoriza a produção em tempo real
- **OEE/Downtime** – Monitoriza a eficiência da produção
- **Web Services** – Conecta com o ERP (Planeaamento de Recursos Empresariais) e outros serviços usando serviços web
- **Track & Trace** – Rastreia produto final, matérias primas e mais
- **SPC** – Coleta e monitoriza informação estatística de controlo do processo
- **Instrument Interface** – Coleta e apresenta a informação a partir de escalas, indicadores e ficheiros
- **Recipe / Changeover Management** – Gere e monitoriza alterações da informação

Na Figura 3.4 estão representadas muitas outras funcionalidades deste *software.*

Figura 3.4: Representação das funcionalidades do Software SCADA Ignition

3.4.1.2 IGSS - Interactive Graphical SCADA System

IGSS é um *software SCADA* da *Schneider Electric*. Este *software SCADA* é genuinamente cliente/servidor para melhorar a performance e fiabilidade. As mudanças no chão de fábrica são reencaminhadas pelo servidor IGSS para as estações de operadores ou clientes logo de imediato. Pode ser expandido de 50 para 400,000 objetos, a partir de uma estação para 50 estações operadoras. Assim, o projeto de supervisão pode começar por uma simples aplicação e ir ganhando dimensão à media que os processos a supervisionar aumentam de complexidade. Com isso pode ser necessário ir adquirindo expansões com um custo suplementar associado. Esse aumento de complexidade e das respetivas expansões/módulos é representado na Figura 3.5:

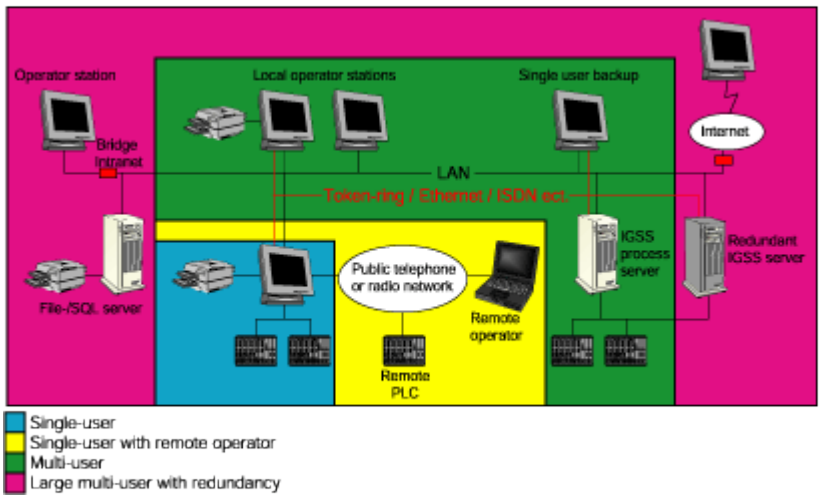


Figura 3.5: Representação do Software SCADA IGSS

A configuração orientada a objetos é uma pedra angular do IGSS. Este tipo de configuração em sistemas de monitorização permite poupança de tempo de desenvolvimento. No caso do IGSS esta configuração fornece consistência no desenvolvimento, flexibilidade, modificação dinâmica da configuração.

Para reduzir tempo de configuração, o IGSS inclui uma característica chamada Group Object. O Group Object é basicamente um *container* o qual é todo ele uma coleção de objetos ou componentes de processo os quais estão logicamente relacionados. Este tipo de *bird's eye view* (olho de falcão) representado por grupo de objetos é usado para copiar todas as configurações ou componentes do mesmo com uma vista para reutilização na mesma configuração IGSS de onde eles se encontram originalmente, ou pela exportação completa de novas configurações.

3.4.1.3 Wincc - Siemens

O *Simatic WinCC* é um *software* da *Siemens* que tem liderado o mercado mundial de *software SCADA* ao longo dos últimos 20 anos. À imagem de outros *software* a escalabilidade está presente neste sistema. Com um pequeno número de estações operadoras no início da fase de planeamento é possível cobrir uma grande gama de requisitos mais tarde como conceitos *web-based* e a conexão a sistemas de informação de alto nível. Outra vantagem é a possibilidade de poder ser integrado com autómatos da mesma marca recorrendo ao *software TIA Portal*. Por exemplo um alarme de um autómato *S7-1500* configurado no *TIA Portal* pode simplesmente ser aplicado à configuração *WinCC* apenas com um botão. O processo histórico do *WinCC* grava informação a partir de qualquer número de sistemas WinCC. Com o *Simatic Information Server* podem ser disponibilizados relatórios online com possibilidade de serem visualizados em dispositivos móveis. Além disso existe a possibilidade destes relatórios serem criados em ferramentas mais familiares, como o *Microsoft Word, Excel ou PowerPoint*. Isto deve-se ao facto de inicialmente o sistema *SCADA WinCC* ter sido desenvolvido para plataformas com o sistema operativo *Windows*. O recurso a dispositivos móveis simplifica a monitorização de processos complexos. A arquitetura do Software está representada na Figura 3.6.

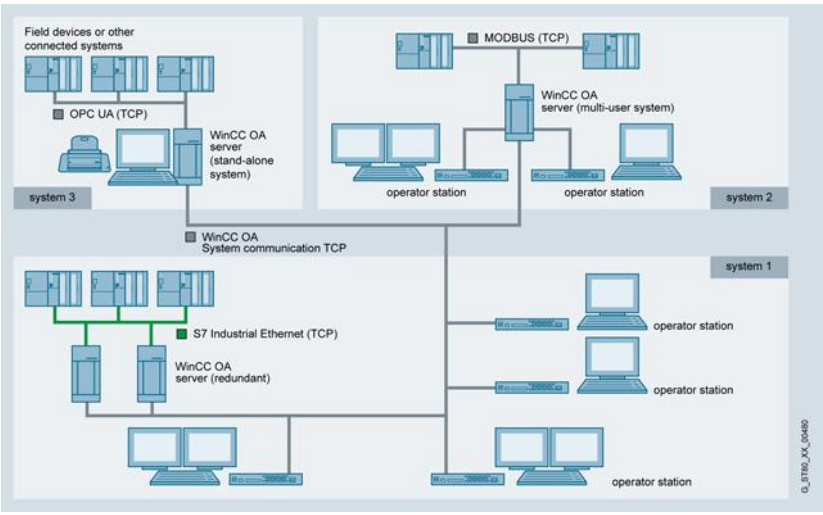


Figura 3.6 - SIMATIC WinCC Open Architecture Add-ons [15]

O Software *WinCC* é um software modular. Algumas expansões possíveis de adquirir estão descritas abaixo:

- *Simatic Process Historian*
- *WinCC Redundancy*

- *WinCC Server*
- *WinCC Telecontrol*
- *WinCC WebUX*
- *WinCC Web Navigator*

3.4.1.4 Movicon

O *Movicon* é uma plataforma de software para criar e correr qualquer tipo de aplicação de supervisão, recolhe dados, análise e controlo, em qualquer indústria ou edifício em contexto de automação. Totalmente baseado em arquitetura *XML* (eXtensible Markup Language) é um software adequado para todo o tipo de aplicações como produção, controlo de processo, infraestruturas, energia, automação de edifícios.

Os projetos podem ser do tipo cliente, servidor, ou web servidor, independentemente da plataforma usada. A arquitetura *Movicon* é completamente baseada em tecnologia XML standard.

Os ficheiros *XML* de um projeto podem ser executados em qualquer plataforma com *Windows 64 bit, Windows 32 bit, Windows CE*. Para segurança os ficheiros do projeto podem ser encriptados. Os projetos do *Movicon* tem relações do tipo *Parent-Children*. O *debugger* pode ser utilizado para projetos locais ou projetos remotos através de uma rede. Este *debugger* pode comunicar de um *PC (Personal Computer) desktop* para dispositivo com sistema embebido (*Win7ES, XPE or Win CE*) através da rede. Através do *debugger* é possível conseguir informação como estatísticas, memória usada etc. É possível também forçar valores de variáveis. O *Movicon* possui também uma vasta libraria de *I/O drivers*, podendo assim configurar rapidamente dispositivos com o mais variado *Hardware*.

Integra tecnologia *SoftPLC* que une os benefícios do *SCADA/HMI* com os ambientes de programação *PLC* de acordo com a norma IEC-61131 [16].

Uma representação de uma solução de monitorização de energia encontra-se representada na Figura 3.7.

Figura 3.7 - Power Consumption Analysis and Energy Efficiency [17]

3.4.2 *PLC* e *RTU*

Como já referido, os sistemas *SCADA* não se cingem apenas ao software que permite a visualização das várias variáveis do processo. Para essa informação chegar aos *software*/interfaces, é preciso as variáveis de interesse no processo em questão serem mensuradas, e essa informação ser transposta para um protocolo conhecido pela rede para poder chegar ao *software*. Esse trabalho é feito pelos *RTU's* ou *PLC's* ou ambos.

RTU, em inglês, *remote terminal unit*, é um dispositivo eletrónico que mede grandezas analógicas ou digitais de um dado processo ou fenómeno e que transmite essa informação para a estação central de monitorização. Este tipo de dispositivo contém *software* que permite converter a informação de entrada em informação de saída segundo o protocolo da rede. Estes dispositivos podem também desempenhar o papel de atuadores locais através de um possível relé que contenham e que seja ativado remotamente para executar qualquer função. Em outros casos podem também conter saídas analógicas[18]. Um exemplo de uma aplicação de um *RTU* está representado na Figura 3.8, em que analisadores de energia estão ligados a uma rede de comunicação.

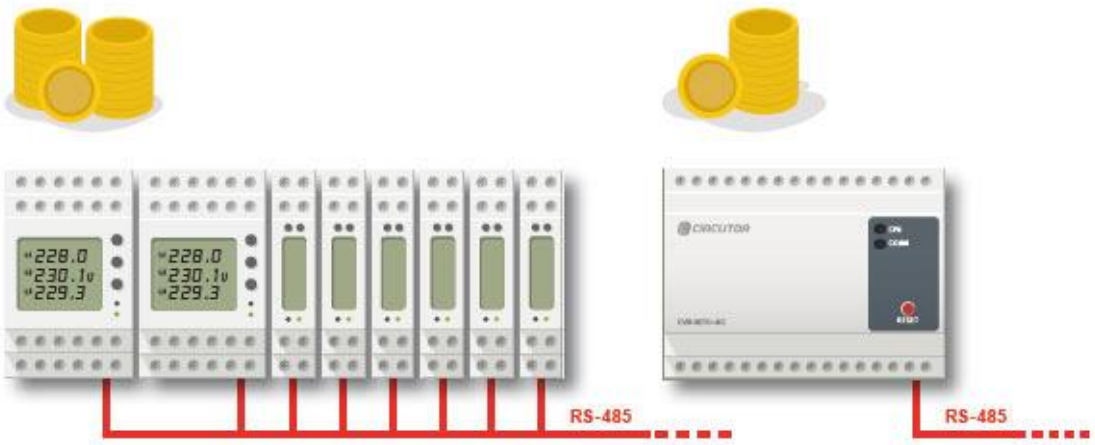


Figura 3.8 – Exemplo de RTU's [19]

PLC (programmable logic controller) ou em português autómato, é um computador especializado usado para controlar processos ou máquinas. Contém memória programável para armazenar instruções e executar funções específicas como controlo on/off, temporização, contagem, sequenciamento, aritmética e manipulação de dados. Projetados para tomar decisões lógicas e fornecer os outputs, os PLC's são usados na sua maioria para controlo de maquinaria ou processos industriais[20]. São fáceis de programar pois a grande maioria pode ser programado recorrendo à linguagem de contactos *Ladder*, que simula a programação de um conjunto de relés, método que era utilizado antes do aparecimento dos *PLC's*.

3.4.2.1 RTU vs PLC

Com a vasta gama de *RTU's* e *PLC's* que se encontram neste momento no mercado, os engenheiros de sistemas *SCADA* e os decisores deste tipo de sistemas enfrentam alguns desafios. Qual a classe destes aparelhos que proporciona a melhor funcionalidade, expansibilidade, e eficiência de custos para um dado sistema *SCADA*? Qual o tipo de unidade que desempenha bem a função não só hoje, mas também durante anos no futuro?

RTU's e *PLC's* compartilham alguns detalhes de arquitetura. No entanto possuem algumas diferenças e cada um tem as suas desvantagens e vantagens.

Muitas industrias e infraestruturas dependem de equipamento localizado em muitos sítios dispersos ao longo de uma grande área geográfica. Os sistemas *SCADA* fornecem monitorização controlo, e automação para melhorar a confiabilidade operacional, reduzir custos com a força de trabalho utilizada, melhorar a qualidade de serviço global (Quality of Service [QoS]), ou conhecer a QoS expectável, ou outros indicadores chave de performance.

Em sistemas *SCADA* os *PLC's* e os *RTU's* desempenham a maioria dos controlos locais. Adquirem a informação local a qual inclui leituras de medidores de pressão temperatura, voltagem, ou outro estado de equipamento, e então atuam localmente e transferem a informação para a central do sistema *SCADA*. Contudo, quando comparados e especificando uma solução para enfrentar os ambientes *SCADA* os *RTU's* e os *PLC's* não são iguais.

Originalmente projetados para substituir os relés lógicos, *PLC's* adquiriram informação analógica e digital através de módulos de *input* e executam um programa em *loop* enquanto analisam os *inputs* e tomam ações baseadas nessa análise. Os *PLC's* mostram uma boa performance em aplicações sequenciais lógicas com grandes quantidades de *I/O* discretos, mas têm uma performance do *CPU* baixa, comunicação pouco flexível e é um sistema de difícil escalabilidade quando é necessário fazer adição de algo mais do que *IO's*.

O típico *PLC* é utilizado numa área local para rápido controlo de variáveis discretas. A utilização de *RTU's* têm-se focado na monitorização remota com controlo, pois têm maior flexibilidade de protocolos e comunicações.

Como resultado, arquitetura *RTU* tende a ter mais poder de processamento, flexibilidade de programação e suporte de comunicação mais amplo do que os *PLC's*.

RTU's e *PLC's* modulares contém *CPU* e *I/O* separados, módulos de comunicação, e suportam a adição de módulos novos [21].

3.5 Protocolos e meios de comunicação industrial

Em vários sistemas como é o caso dos sistemas *SCADA*, é necessário que aparelhos diferentes comuniquem entre si. Para esta comunicação é necessária a existência de regras comuns. Um protocolo, que não é mais que uma série de regras que permitem a troca de mensagens e a sua sincronização.

No caso específico da indústria a necessidade da maximização da produtividade e controlo de qualidade levou à comunicação entre o chão de fábrica e outros pontos da fábrica através de redes de comunicação de dados. *HMI's, PLC's,* sensores, precisam de ser conectados de forma eficiente e escalável [22].

"Historicamente, muitos componentes industriais têm sido conectados através de diferentes protocolos série como *CAN, Modbus, Profibus e CC-Link*. Ultimamente o *Ethernet* industrial tem ganho popularidade, tornando-se mais presente e oferecendo alta velocidade, distância de conexão e conexão de mais nós."[22]

Serão abordados seguidamente os protocolos usados neste trabalho.

3.5.1 Modbus

O protocolo *Modbus* é uma estrutura de mensagens desenvolvida pela *Modicon* em 1979. Este protocolo é usado para estabelecer ligações do tipo *master/slave* ou cliente/servidor entre dispositivos. É um dos protocolos mais utlizados em automação industrial. Sendo um protocolo largamente aceite e utilizado por dispositivos industriais de vários fabricantes faz com que seja fácil de usar e fiável. É fortemente utilizado em *Building Management Systems* (BMS) e *Industrial Automation Systems (IAS)* [23].

Existem dois tipos de *Modbus*: o *Modbus RTU* e o Modbus TCP/IP.

3.5.2 Modbus RTU

Este protocolo utiliza comunicação série (*RS232 e RS485*), propõe uma arquitetura *master/slave* e um conjunto de mensagens que todos os equipamentos sabem interpretar. Para detetar erros de transmissão este protocolo propõe o cálculo e transmissão do *CRC (Cyclic-Redundant Checksum)* no final de cada mensagem *Modbus.* O CRC consiste no envio de dois bytes, esses bytes são calculados em função da mensagem enviada de acordo com o algoritmo *CRC16*. Os equipamentos que receberem esta mensagem calculam localmente o *CRC16* da mensagem recebida. Se o *CRC16* calculado localmente for igual ao enviado na mensagem, o aparelho assume que a transmissão ocorreu sem erros.

Este protocolo impõe uma estrutura às mensagens trocadas entre dois ou mais aparelhos constituintes de uma rede. Um desses aparelhos terá de ser o master, e todos os outros aparelhos serão os *slaves*. O master é o aparelho responsável por realizar pedidos aos *slaves* e estes apenas respondem aos pedidos que lhes são feitos. A comunicação é série, *RS232 e RS485*. À semelhança deste protocolo, existe o *ModBus Ascii*, que difere em pequenos pontos do *RTU* mas o mais utilizado por aparelhos industriais é o *RTU*.

Algumas vantagens deste protocolo são as seguintes: fácil implementação comparando com protocolos atuais, necessidade de pouca memória para a sua implementação num aparelho programável (cerca de 2Kb).

No entanto também existem obviamente algumas desvantagens como por exemplo: o número de *slaves* está limitado a menos de 255 (2^{8-1}) (alguns valores são reservados para o sistema), apenas pode ser implementado em redes de comunicação série o que limita a distância de comunicação e mesmo o número de nós, todos os aparelhos têm de ser configurados com o mesmo tipo de comunicação (baud rate, paridade, etc..).

3.5.3 Modbus TCP/IP

O Modbus TCP/IP é uma variante do *Modbus* e tem como meio de comunicação redes de *Ethernet* utilizando o nível 7 do modelo de OSI, que proporciona uma comunicação do tipo cliente/servidor entre dispositivos. Através de uma rede de internet ou intranet e utilizando o protocolo *TCP/IP* é possível enviar e receber mensagens deste protocolo. No *Modbus TCP/IP* uma conexão é facilmente reconhecida ao nível do protocolo, e uma única conexão pode operar várias transações. No entanto o protocolo *TCP/IP* permite várias conexões ao mesmo tempo, por isso na maioria dos casos é escolhida a opção de conectar consecutivamente ou reusar a mesma conexão [24].

Este protocolo utiliza a porta 502 do protocolo *TCP/IP.* Feita uma conexão entre cliente e servidor, o servidor responde aos pedidos do cliente até que o cliente encerre a ligação.

A grande vantagem de utilizar o *Modbus TCP/IP*, é o aproveitamento das redes de *Ethernet* que normalmente já existem em meio industrial podendo assim ligar o chão de fábrica ao nível dos gabinetes. Sendo mais imune a ruído o *TCP/IP Ethernet*, permite aumentar a distância em relação à comunicação série *RS485*.

4

Solução e Implementação Propostas

4 Solução e Implementação propostas

Depois de analisados os consumos de energia na Colep Portugal, utilizando dados que foram fornecidos pelo *Energy Manager* da empresa que foram o resultado de uma auditoria realizada em 2012, foi tomada a decisão de atuar sobre o consumo de energia elétrica. Esta energia representa mais de metade do custo da energia (62,3%) e em termos energéticos quando comparados em tep representa uma fatia ainda maior, cerca de 66,5%.

Assim, a solução proposta foi o desenvolvimento de um sistema de monitorização de energia elétrica. Para implementação do sistema na empresa, e tendo em conta que é pretendido que este sistema:

- Concentre os valores de contagens de energia numa única base de dados
- Não necessite de intervenção humana constantemente
- Frequência de amostragem elevada
- Baixo custo e seja versátil para os vários tipos de contador de energia elétrica

Foi pensada uma arquitetura dividida em **3 partes** como representado na Figura 4.1:

- O **baixo nível/ RTU**: chão de fábrica, onde se encontram os contadores
- O **médio nível/PLC**: Hardware/Software responsável pela aquisição de dados
- **Alto nível/HMI**: Onde se encontra a base de dados e são guardadas as contagens

Figura 4.1 Níveis da arquitetura

O baixo nível ou zona *RTU* é onde se encontram os contadores de energia elétrica. Estes contadores podem ter comunicação ou não. Os contadores que não possuem comunicação serão propostos a ser substituídos. Quanto aos que têm comunicação, esta pode ser feita de duas formas dependendo do modelo. Existe comunicação inteligente, que se resume ao

protocolo *Modbus RTU* no modo *slave* por parte dos contadores, sobre a comunicação *RS485*. Alguns contadores possuem também comunicação inteligente sobre *RS485* com protocolos proprietários da marca. Outro tipo de comunicação apresentado pelos contadores de energia elétrica da fábrica é a chamada a comunicação por impulso elétrico. Este tipo de comunicação resume-se a um relé interno do contador de energia que fecha um contacto durante um determinado tempo à contagem de uma determinada quantidade de energia. Assim, dependendo da marca do contador de energia elétrica, é possível medir remotamente a energia consumida na instalação desse contador.

No médio nível ou zona PLC serão concentrados os valores das contagens dos vários contadores e reunidos num único aparelho/sistema. É neste nível que os dados vão ser passados para um protocolo que possa ser enviado para o HMI.

No HMI haverá a interface com o utilizador e uma base de dados onde serão guardados os históricos de consumos de energia elétrica.

A arquitetura proposta é apresentada na Figura 4.2.

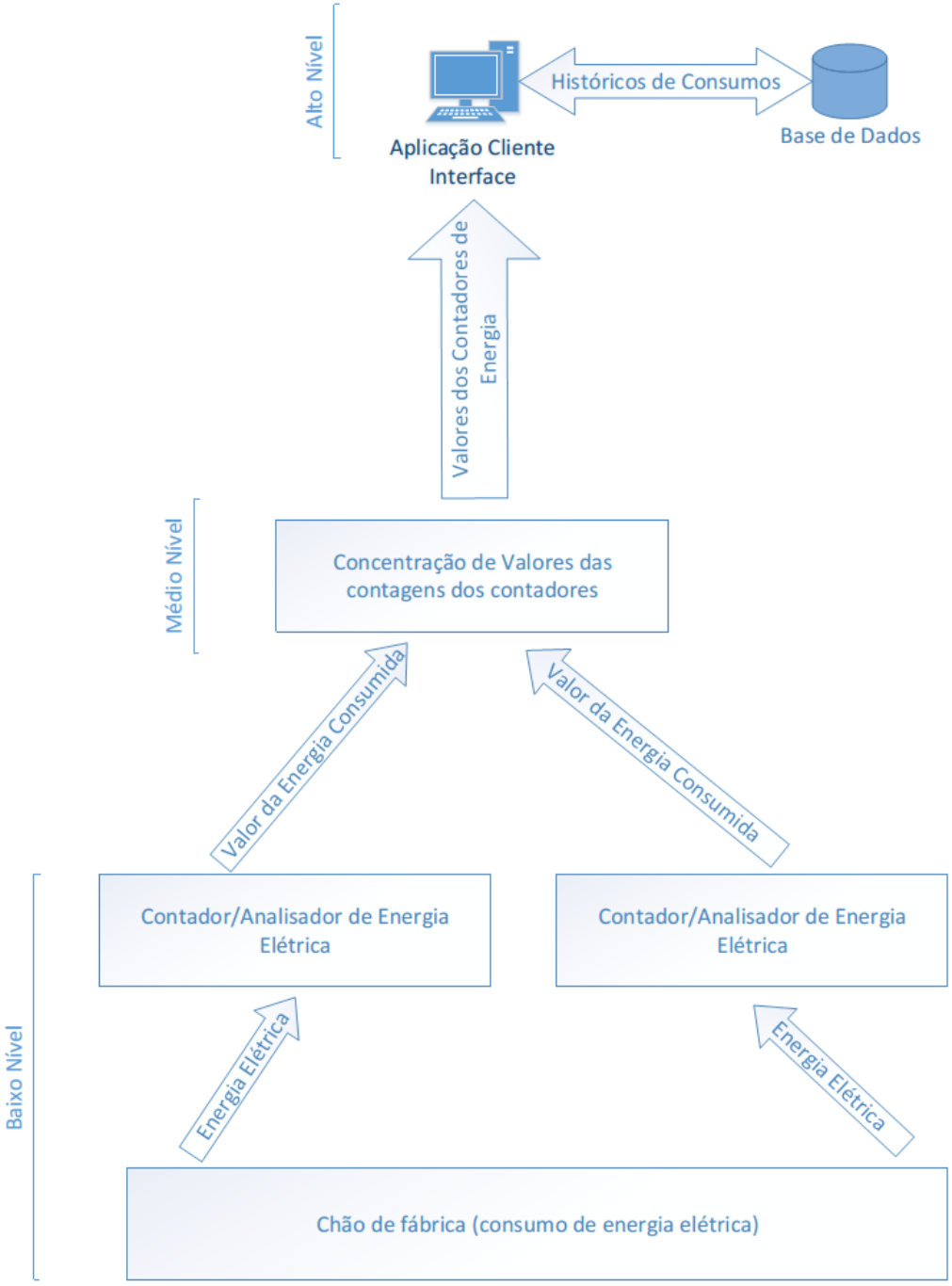


Figura 4.2 Arquitetura conceptual

4.1 Implementação proposta

A escolha de protocolos e equipamentos para a solução proposta está representada na Figura 4.3.

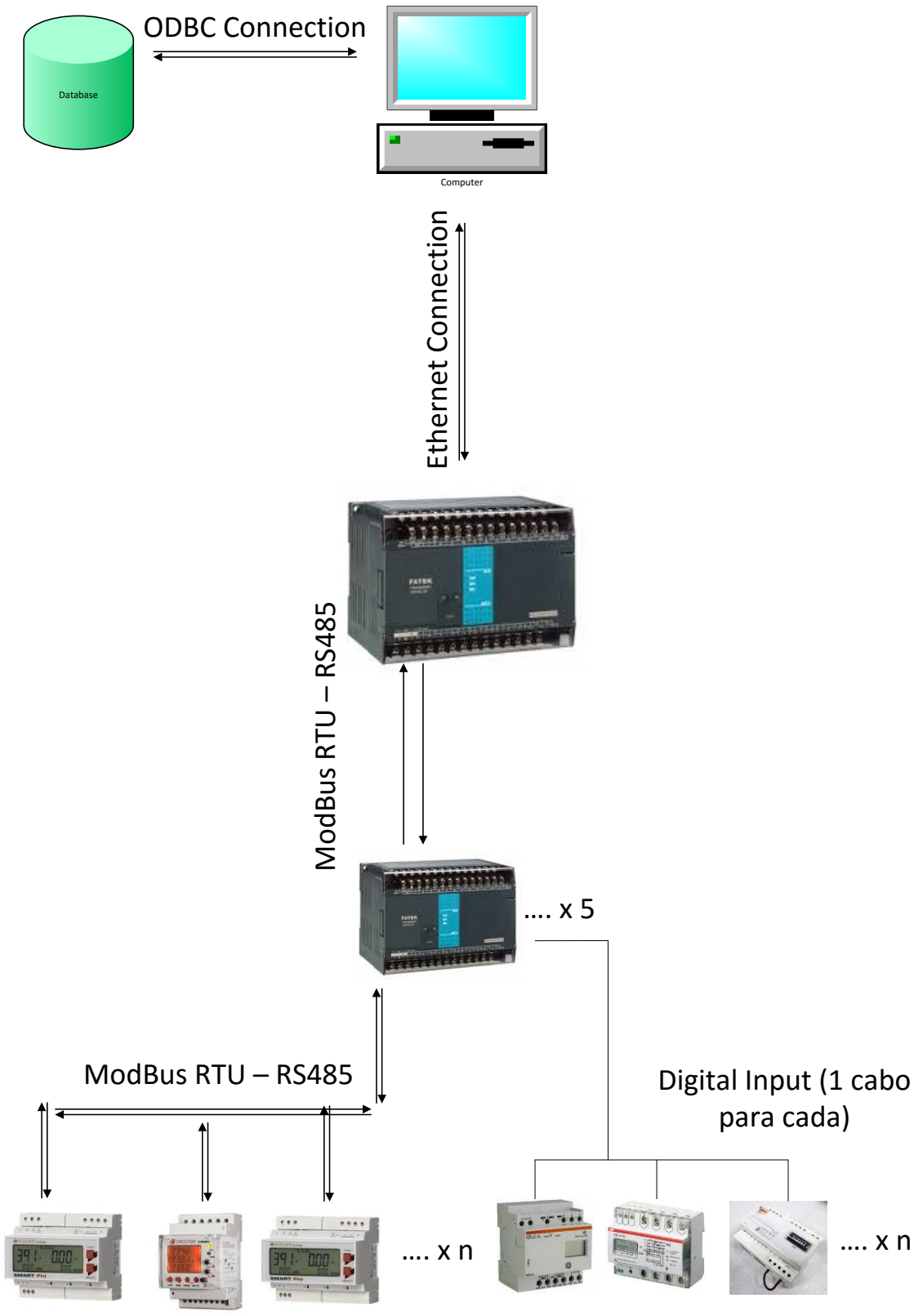


Figura 4.3 Solução de Hardware e Software para a arquitetura

4.1.1 Zona RTU

Para encontrar uma solução para a aquisição dos dados dos contadores de energia elétrica da fábrica, foi necessário fazer um levantamento dos contadores existentes. Depois de feito esse levantamento foi reunida a informação quanto à marca e modelo do equipamento, podendo assim consultar os manuais para se proceder à avaliação das características quanto à comunicação possível. As existências de contadores e as informações reunidas estão representadas na Tabela 4.1.

Tabela 4.1 – Existências de Contadores de Energia Elétrica na fábrica

MARCA	MODELO	Comunicação	Número de contadores	Total
MERLIN GERIN	**ME4 ZRT**	PULSE	7	65
REGULADORA	**-**	0	4	
MERLIN GERIN	**CER**	PULSE	24	
MERLIN GERIN	**PM 700**	0	1	
MERLIN GERIN	**PM 710**	RS485	3	
ABB	**OD 4110**	PULSE	6	
SIEMENS	**D11R**	0	8	
DUCATI	**SMART PIU**	RS485	9	
CIRCUTOR	**CVM MINI**	RS485	1	
SCHNEIDER ELECTRIC	**A9 MEM 3210**	PULSE	1	
SCHNEIDER ELECTRIC	**ME4 ZRT**	PULSE	1	

Os contadores com um "0" no campo "Comunicação" não contêm qualquer tipo de comunicação que possibilite a mediação da energia de forma remota. Assim estes contadores não foram considerados na arquitetura e mais à frente neste trabalho será sugerida a sua substituição. Os contadores que contêm "PULSE" na coluna referente à "Comunicação" contém um relé interno que fecha um contacto um determinado número de vezes e durante um espaço de tempo (dependendo do modelo e da marca do contador) por cada KWh de energia consumida e consequente medida no contador. Estes aparelhos contêm dois conectores que ligam as extremidades desse contacto que é fechado pelo relé. Assim, alimentado com uma certa diferença de potencial esses conectores, podemos ter um impulso elétrico sempre que o contacto é fechado. Por último os contadores a que corresponde o valor "RS485" na coluna da "Comunicação" são os contadores que possuem comunicação inteligente. Todos estes contadores possuem a capacidade de comunicar através do protocolo *Modbus RTU* no modo *slave*. Alguns destes contadores também possuem a capacidade de utilizar outros protocolos como *Modbus ASCII* e até protocolos proprietários. No entanto dos vários protocolos que os contadores possuíam, todos tinham em comum o protocolo *Modbus RTU*, e foi este o utilizado como se vai verificar mais à frente no documento. O meio físico desta comunicação resume-se à comunicação série *RS485*.

Em suma, os vários modelos e marcas de contadores que foram encontrados no chão de fábrica podem ser divididos em três tipos:

- Com comunicação **RS485 Modbus RTU**

- Com comunicação por **impulso elétrico**
- Sem qualquer tipo de comunicação

Deste modo a decisão para a arquitetura da zona RTU foi utilizar o sinal de impulso elétrico em conjunto com o protocolo *Modbus RTU* sobre *RS485*, descartando os contadores sem comunicação. Assim, para realizar este projeto, foi necessário pensar numa solução de aquisição de dados (**PLC**) que aceite estes 2 tipos de comunicação. É essa arquitetura que vai ser abordada na secção seguinte

4.1.2 Zona PLC

É neste nível que a informação proveniente dos contadores tem de ser recebida e enviada para o HMI para que este possa trabalha-la. Assim tem de ser selecionado o software/hardware que possa receber a informação vinda dos contadores e enviá-la para o HMI. Ora como na secção anterior foi constatado, a arquitetura deste nível tem de receber a informação dos contadores em forma de impulso elétrico e protocolo *Modbus RTU* sobre *RS485*. Quanto aos contadores sem qualquer tipo de comunicação foram descartados nesta arquitetura e será proposta a sua substituição. Depois de analisadas as soluções oferecidas pelo mercado para este tipo de comunicação foi escolhida a solução de um autómato *Fatek*. Esta solução é versátil, de programação fácil e é uma solução barata quando comparada com soluções como *Remote IO*.

Tendo em conta a dimensão da empresa e dispersão dos contadores ao longo dos vários setores da fábrica, e para evitar demasiados metros de cabo de comunicação, foi escolhido um modelo em que existe um autómato local que recolhe os valores de uma determinada área. Estes autómatos locais reportam os dados para um autómato global que por sua vez envia os dados para o HMI. Desta forma foi necessário fazer o levantamento geográfico dos contadores, o seu tipo de comunicação e dividir em zonas como mostra a Figura 4.4.

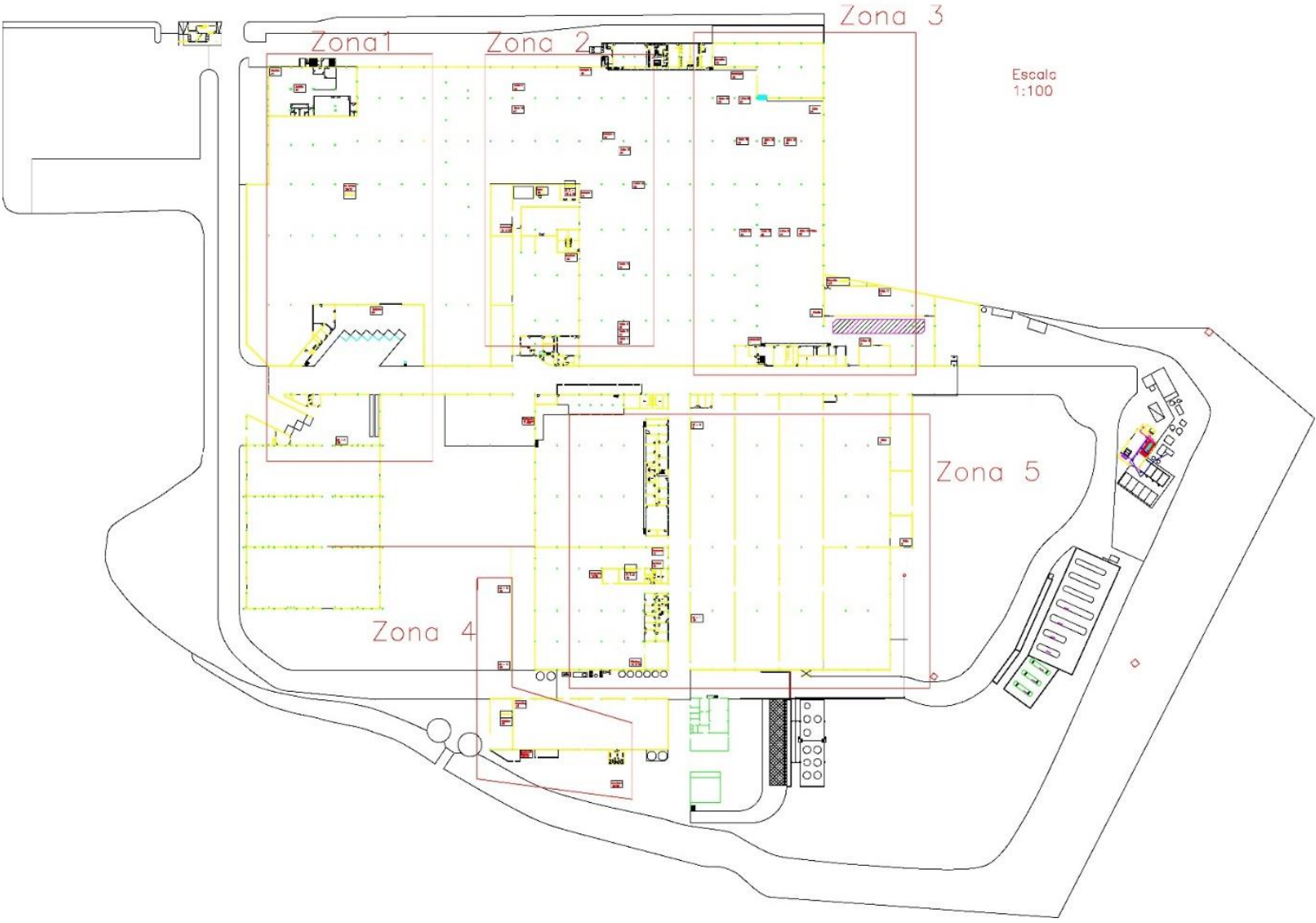


Figura 4.4 - Divisão da fábrica em zonas

Feita a divisão do chão de fábrica em zonas tendo em conta a proximidade dos contadores e o seu tipo de comunicação, passaram a existir cinco zonas como mostra a Figura 4.4. Nestas zonas foram contabilizados o número de contadores e o seu tipo de comunicação como representado na Tabela 4.2.

Tabela 4.2 – Tipos e quantidade de contadores por zona

Zona	Contadores com comunicação inteligente	Contadores com comunicação de impulso elétrico	Sem comunicação	Total
1	0	5	1	6
2	2	14	5	21
3	10	4	4	18
4	0	8	0	8
5	1	8	3	12

Com esta contabilização dos contadores por zona, é possível fazer a escolha do autómato que mais se adequa a cada local. O critério a ter em conta será o tipo de comunicação de cada contador e a quantidade destes. Nas secções seguintes serão explicadas as escolhas para cada um dos autómatos locais.

Para melhor perceber esta arquitetura do sistema por zona e as escolhas efetuadas, na seguinte Figura 4.5 é possível ver um esquema do tipo e número de contadores por cada zona da empresa.

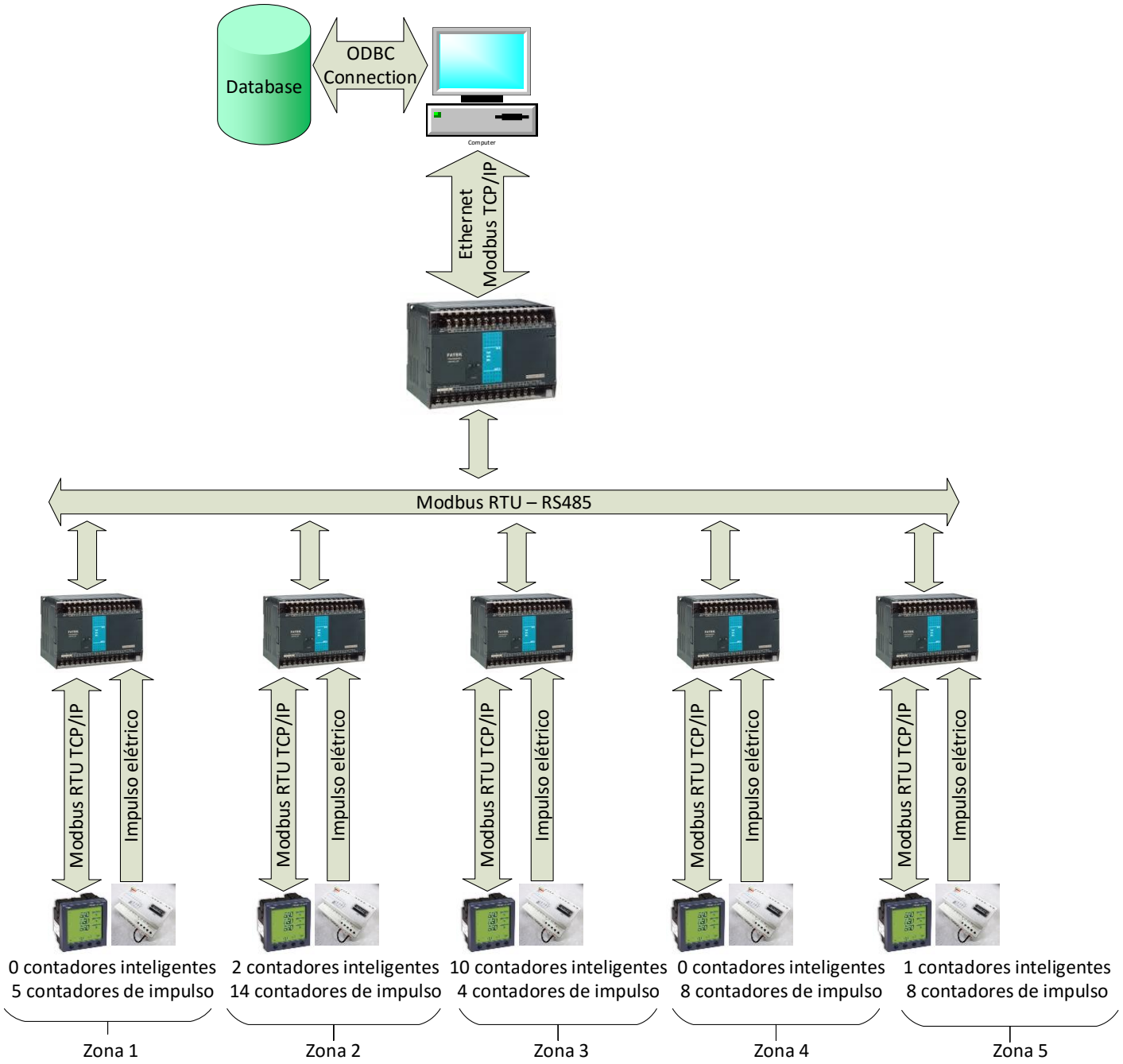


Figura 4.5 – Tipos de contadores por zona

4.1.2.1 Zona 1

Para a zona 1 como podemos verificar na Tabela 4.2 e Figura 4.1 o autómato necessário precisa de acomodar a informação de cinco contadores com comunicação por impulso elétrico e um

sem comunicação que como já referido este tipo de contadores terá a sua substituição proposta neste documento. Como já referido os autómatos locais terão de reportar a informação a um autómato principal que por sua vez envia a informação para o HMI. Ora esta comunicação entre os autómatos locais e o autómato principal faz-se através de uma rede RS485 segundo o protocolo *Modbus RTU* como representado na Figura 4.3. Por este motivo estes autómatos locais precisam de uma porta de comunicação RS485. Contudo estes autómatos também comunicam através de uma rede deste tipo com os contadores que possuem comunicação inteligente. No entanto esta rede RS485 não é a mesma que a que liga os autómatos locais ao autómato principal. Desta forma, o autómato local terá de ter duas portas de comunicação RS485 independentes. Ora no caso particular desta zona, não existe nenhum contador com comunicação inteligente, mas tendo em conta que um dos contadores não possuir comunicação alguma e a sua substituição ser proposta e permitir que esta arquitetura seja escalável para mais contadores e até para contadores de outros tipos de energia no futuro, todos os autómatos locais terão as duas portas de comunicação. Posto isto o autómato escolhido foi da marca *Fatek*, possui oito entradas digitais (para acomodar a informação dos contadores com comunicação por impulso elétrico) e foi adquirida uma expansão superior com duas portas de comunicação RS485.

4.1.2.2 Zona 2

Para a zona 2 o autómato local tem de receber informação de 2 contadores com comunicação inteligente, 14 contadores com comunicação por impulso elétrico e ainda possui 5 contadores que não possuem qualquer tipo de comunicação. Ora tendo em conta as 14 entradas digitais necessárias e tendo em conta que poderá haver possibilidade de serem necessárias mais entradas digitais seja pelos 5 contadores a serem substituídos nesta zona ou seja por uma questão de escalabilidade do sistema, foi escolhido um autómato idêntico ao da zona 1, mas com 20 entradas digitais e uma expansão igual.

4.1.2.3 Zona 3

Na zona 3 existem 10 contadores inteligentes 4 com comunicação por impulso elétrico e 4 sem comunicação possível. Á imagem da zona 1 o autómato escolhido foi um autómato com 8 entradas digitais e como sempre com a expansão com duas portas de comunicação RS485. Assim mesmo que os 4 contadores desta zona que não possuem comunicação sejam substituídos por contadores com comunicação por impulso elétrico, este autómato tem entradas disponíveis para o efeito.

4.1.2.4 Zona 4

Nesta zona da fábrica existem 8 contadores de energia elétrica. Todos estes contadores possuem comunicação por impulso. O autómato escolhido para esta zona possui 12 entradas digitais. Apesar de serem só necessárias 8 entradas digitais, optou-se por sobre dimensionar para acomodar possíveis expansões de rede a outros contadores de outros tipos de energia ou até mesmo a um aumento do número de contadores de energia elétrica nesta zona. A expansão escolhida foi a mesma dos outros autómatos. Mais uma vez há um sobre dimensionamento pois seria preciso apenas uma porta de comunicação RS485, mas optou-se por escolher duas portas devido a mais uma vez, tornar a escalabilidade do sistema um dos seus pontos fortes.

4.1.2.5 Zona 5

Por fim a zona 5 como é possível verificar na Tabela 4.2 esta zona contém 1 cotador de energia elétrica com comunicação inteligente, 8 contadores com comunicação por impulso elétrico e 3 contadores sem comunicação. À imagem da zona 4 foi escolhido um autómato com 12 entradas

digitais. Mais uma vez sobre dimensionou-se tendo em conta futuras necessidades de expansão de rede, ou até mesmo à substituição dos contadores sem comunicação por contadores que apenas possuam comunicação por impulso e não comunicação inteligente.

4.1.2.6 Autómato central

Para reunir todos os valores de contagens que os contadores locais registam é necessário um autómato central como exemplificado na arquitetura proposta na Figura 4.3. como a função principal deste autómato era receber os dados da rede de comunicação RS485 Modbus RTU que o ligava aos autómatos locais, e enviar essa informação para a rede de Ethernet da empresa. Desta forma foi escolhido um autómato com o mínimo de entradas digitais possíveis, com 6 entradas digitais, e com uma expansão lateral que possui uma porta de Ethernet e uma RS485. Mais à frente neste documento vai ser mostrado que este autómato foi adquirido e foi montada uma rede de teste.

4.1.2.7 Autómatos necessários

Da análise feita nas secções anteriores foi deita a seleção dos autómatos necessários para esta arquitetura. Na Tabela 4.3 pode verificar-se a escolha de autómatos para cada zona:

Tabela 4.3 – Escolha de autómatos e expansões por zona

Zona	Autómato	Expansão	Entradas Digitais	Portas de comunicação
1	Fatek FBs-14MAR2-AC	Fatek FBs-CB55	8	2*RS485
2	Fatek FBs-32MAR2-AC	Fatek FBs-CB55	20	2*RS485
3	Fatek FBs-14MAR2-AC	Fatek FBs-CB55	8	2*RS485
4	Fatek FBs-20MAR2-AC	Fatek FBs-CB55	12	2*RS485
5	Fatek FBs-20MAR2-AC	Fatek FBs-CB55	12	2*RS485
Principal	Fatek FBs-10MCR2-AC	Fatek FBs-CM55E	6	RS485 + Ethernet

4.1.3 HMI

Este nível como já referido é o responsável por registar e apresentar os dados das contagens de energia. É neste nível da arquitetura que se encontra o cliente/master da camada PLC e que efetua pedidos dos valores pretendidos. De grosso modo este nível pode ser dividido em duas partes: o pedido dos dados à camada PLC e o seu processamento, e o armazenamento dos dados.

O pedido dos dados e comunicação inerente com a camada PLC é feito por um computador comum gerido por uma aplicação concebida para o efeito. Esta aplicação foi desenvolvida em Visual Basic .NET, e foi eleita devido à familiarização com esta linguagem e ao facto de esta linguagem ser uma linguagem de programação de desenvolvimento rápido de aplicações.

Já o armazenamento dos dados é feito numa base de dados Microsoft Access. Esta base de dados foi escolhida pois já é utilizada na empresa para fazer o histórico dos consumos mensais pelo Energy Manager.

Nas secções seguintes estão explicadas as duas partes constituintes desta arquitetura.

4.1.3.1 Programa Visual Basic .Net

O programa responsável por realizar os pedidos foi desenvolvido na linguagem de programação Visual Basic .Net. esta linguagem foi escolhida devido à familiarização com a mesma e à rapidez

do desenvolvimento de ampliações. O ambiente de desenvolvimento foi o IDE da Microsoft, *Visual Studio 2013*. Este IDE tem grandes facilidades de desenvolvimento ao nível do *debug* do programa a desenvolver, assim como sugestão de escrita.

A aplicação desenvolvida tem duas funções principais: a comunicação com a camada *PLC* para o pedido dos dados segundo o protocolo *Modbus TCP/IP* sobre a ligação física de *Ethernet*, e a comunicação com a base dados *Access* segundo a ligação *ODBC*.

Para a ligação à camada *PLC* foi utilizado o protocolo *Modbus TCP/IP*. Para esta ligação haviam várias possibilidades, entre as quais, o protocolo proprietário da *Fatek* e a utilização do serviço de *OPC Server* disponibilizado pela *Fatek*. Ora estas soluções iriam tornar este programa dependente da marca do autómato numa futura expansão de rede, o que iria afetar o leque de escolhas numa expansão futura. Desta forma foi optada por a solução do protocolo *Modbus TCP/IP* sobre *Ethernet*, o que torna este programa independente do tipo de aparelho da camada *PLC*, exigindo apenas que estes possam comunicar segundo este protocolo que é standard e aberto. Ora esta opção teve uma contrapartida. O trabalho de desenvolvimento foi maior pois com a utilização de um *OPC*, a comunicação ficava a cargo deste e o programa apenas teria de invocar os comandos para a troca de dados. Como o protocolo escolhido foi o *Modbus TCP/IP*, este teve de ser implementado no desenvolvimento da aplicação. A abrangência do programa devido à utilização de um protocolo standard, acrescenta muito valor pois não o torna dependente de uma marca ou tipo de aparelho, mas teve um custo na dificuldade e tempo de desenvolvimento na aplicação.

Para a ligação à base dados foi utilizada uma ligação do tipo *ODBC* por a familiarização com esta. Além disso não houve interesse por parte do Energy Manager de possuir uma base de dados remota, mas sim no mesmo computador da aplicação. A linguagem para a comunicação com a base de dados que foi escolhida foi SQL. Esta linguagem possui uma série de comandos que permitem manipular os registos na base dados.

4.2 Implementação de uma rede de teste

Foi implementada na empresa uma rede de teste para poder averiguar a validade da arquitetura desenhada. Esta rede de teste teve o objetivo de testar a capacidade de recolher valores dos dois tipos de contadores, os contadores com comunicação inteligente e os contadores com comunicação por impulsos elétricos. Os dois contadores testados foram:

- Contador de marca e modelo ***ABB OD4110*** responsável, pela contagem da energia do ***Compressor GA 90 VSD***
- Contador da marca e modelo ***Ducati Smart PIU*** que faz a contagem da ***Linha 34***

Como irá ser possível verificar mais à frente neste documento, na Tabela A.1 do Anexo A, o contador do ***Compressor GA 90 VSD,*** possui comunicação por impulso elétrico, e o contador da ***Linha 34***, possui comunicação inteligente segundo o protocolo Modbus RTU sobre o meio físico RS485.

Por conveniência ao Energy Manager da empresa, estes foram os dois contadores escolhidos para a validação da arquitetura.

Para poder ler os valores dos contadores foram necessárias duas coisas. Primeiro saber construir o protocolo *Modbus RTU* e como este funciona. Essa explicação está nos anexos na secção. Em segundo, é necessário consultar o *Datasheet* do aparelho para poder saber qual a posição de memória a que corresponde o valor a ser lido.

4.2.1 Comunicação inteligente (*Modbus RTU RS485*) VS comunicação por impulso (Pulse output)

A comunicação inteligente é preferível à comunicação por impulso elétrico, pois apenas é necessário realizar um pedido e o contador responde o valor que lhe foi pedido pelo master. Enquanto na comunicação por impulso elétrico, é necessário fazer uma calibração inicial onde o master tem de ter o valor atual da contagem do contador para depois poder incrementar e assim o valor estar sincronizado com o valor do contador. Além disso tem de se ter em conta que o contador chegando ao limite da contagem volta a zero, por isso o master também terá de prever essa situação. Uma outra desvantagem será a maior quantidade de cabos, pois para cada contador de comunicação por impulso elétrico terá o seu próprio par de cabos, enquanto os que comunicarem através da rede RS485 podem estar "pendurados" numa única rede de RS485 até um limite de 255 aparelhos.

Assim, sempre que um contador possuir comunicação inteligente está será escolhida e não a comunicação por impulso elétrico.

4.2.2 ABB OD4110

Este contador da marca ABB, é o aparelho responsável pela contagem da energia do Compressor GA 90 VSD. Consultando o seu manual, é possível verificar que este não possui comunicação inteligente. No entanto possui um relé interno (Pulse output) que ao ser alimentado por uma tensão este emite um impulso elétrico que resulta do fecho deste relé interno. Neste caso específico, o relé fecha 100 vezes a cada kWh. Do manual deste contador é possível também obter outras informações à cerca do tipo de alimentação que pode ser feita a este relé. A tensão de alimentação pode variar de 5V a 40V em DC (corrente contínua), com um máximo de 100mA de corrente. A duração do impulso elétrico é de cerca de 100ms ± 2,5ms [25].

Em suma, a maneira de poder retirar o valor da contagem de energia deste contador passa por alimentar o relé interno com uma tensão entre 5V e 40V DC e com um máximo de 100mA de corrente. O impulso resultante terá uma largura de 100ms ± 2,5ms.

4.2.3 Ducati Smart PIU

Este contador é o responsável pela contagem da energia da linha 34. Trata-se de um contador ca marca Ducati e possui comunicação inteligente. Recorrendo ao manual de especificações técnicas é possível verificar que este contador possui várias maneiras possíveis de comunicação. Possui comunicação por impulso e comunicação inteligente sobre RS485 com dois protocolos possíveis. O protocolo *Modbus RTU*, e um protocolo proprietário Ducati ASCII [26].

Como o contador possui comunicação inteligente foi esta a escolhida para a comunicação. Quanto ao protocolo, o escolhido foi o *Modbus RTU* pois é standard enquanto que o protocolo proprietário iria exigir uma aprendizagem com o respetivo tempo e além disso seria apenas útil para os contadores desta marca, o que iria tornar o trabalho bem mais complexo.

4.2.4 Configuração do *Modbus RTU* no contador de energia inteligente

Como é possível verificar na Tabela A.1 do Anexo A, quando foi feito o levantamento dos contadores da empresa, todos foram identificados com um número correspondente. Ora para facilitar e aproveitar esse trabalho efetuado e tendo em conta que o número máximo é 66, os contadores com comunicação inteligente vão receber como ID de *slave* na rede o número a que correspondem na planta.

Quanto à parametrização da comunicação série, foi escolhida a velocidade de transmissão de dados de 9600 bits/s, 8 bits de dados, com 1 stop bit e sem bit paridade. Estes valores para a parametrização da comunicação série foram os escolhidos pois todos os contadores inteligentes existentes na fábrica, podem ser parametrizados desta forma, que é a mais usual nas comunicações séries utilizadas na indústria.

No caso particular do contador Ducati Smart Più, representado na Figura 4.6, da linha 34, este na planta corresponde ao número 41 da planta representada no Anexo B.1. Ou seja, o número do *slave* na rede que o vai corresponder é o número 41. Quanto à configuração da ligação série RS85, foi necessário consultar o seu *datasheet* para o poder parametrizar. No parágrafo seguinte vai ser explicado em detalhe o procedimento.

Consultando o manual do aparelho[27], foi possível saber como configurar o aparelho para este passar a utilizar o protocolo *Modbus RTU*, assim como, configurar a ligação série com os parâmetros seguintes: uma velocidade de transmissão de 9600 bits/s; 8 bits de dados; 1 stop bit; e sem bit de paridade. Na secção 6.4.13 deste manual, foi possível saber como fazer esta configuração do aparelho representado na Figura 4.6.

Figura 4.6 – Ducati SMART PiÙ [28]

Depois de definidos os parâmetros da comunicação série, foi necessário saber as posições de memória com os valores de interesse para serem lidas pelo autómato. Nesta rede de teste, o objetivo era apenas ler o valor da potência consumida. Para isso teve-se de recorrer ao manual de endereços *Modbus RTU* deste aparelho. Nesse manual, o valor da energia consumida está na posição de memória 106, descrito como "*Three-phase equivalent active energy*" e é considerado

um "*Unsigned Long*". Segundo o fabricante esta designação significa que se trata de um número binário constituído de duas "*words*" (32 bits), e sem sinal [27].

Desta forma, quando o autómato for programado, este terá de fazer o pedido Modbus RTU da posição de memória 106 e a posição seguinte.

4.2.5 Pulse Output no contador ABB OD4110

Quanto ao contador ABB OD4110, representado na Figura 4.7, e como já referido na secção 4.2.2, este não possui comunicação inteligente e terá de ser usada a comunicação por impulso. Consultando o manual deste aparelho [25], é possível verificar na secção que este contador possui uma saída de impulsos com uma frequência de 100 impulsos/kWh, com uma área de ligação de 0,2 mm^2 a 2,5 mm^2, com uma corrente máxima de 100 mA e com uma largura do impulso elétrico 100 ms.

Figura 4.7 –Contador de Energia ABB OD4110 [25]

Tendo em conta o sinal de output elétrico este pode ser considerado uma saída digital, e assim cada impulso de output pode ser contabilizado por forma a fazer a contagem de energia deste contador.

4.2.6 Escolha e Programação do autómato

Tendo em conta os dois tipos de comunicação descritos nas duas secções anteriores, foi feita a seleção do autómato para esta rede de teste. A marca escolhida foi a marca Fatek pois trata-se de uma marca bastante barata no que toca ao mercado de autómatos e tendo em conta que uma falha do sistema de monitorização não será crítica para a laboração da empresa, não há necessidade de investir dinheiro num aparelho topo de gama como é o caso dos autómatos Siemens. Desta forma o autómato escolhido foi o Fatek FBs-10MCR2-AC pelo preço de €134,24, e uma expansão de 2 portas RS485 e uma porta Ethernet, com a designação FBs-CM55E e pelo preço de €189,52.

O autómato escolhido tem 6 entradas digitais, uma das quais ver ser utilizada para receber o sinal de impulso do contador ABB OD4110 do compressor GA 90 VSD, e a expansão possui uma porta RS485 (Port 3), que será utilizada para comunicar com o contador inteligente *Ducati Smart Più* através do protocolo *Modbus RTU*. O autómato e a respetiva expansão encontram-se representados na Figura 4.8, já instalados no quadro elétrico.

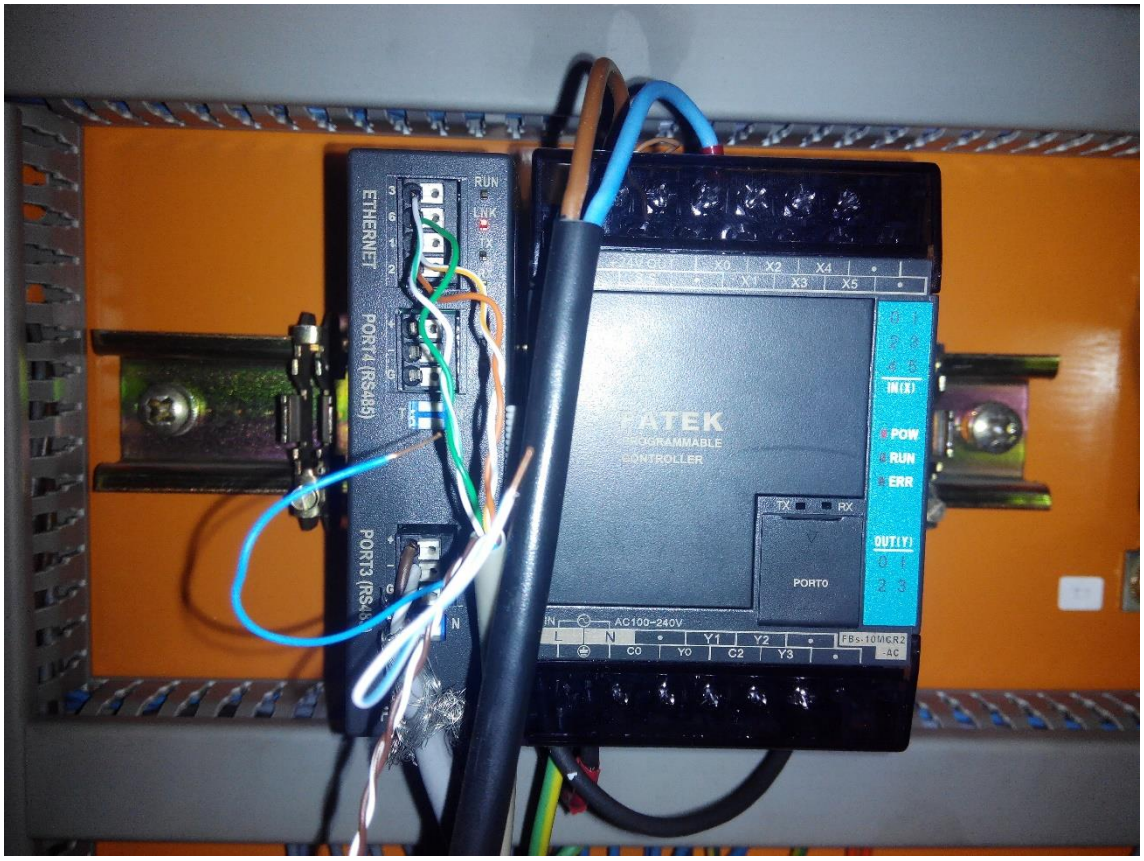


Figura 4.8 – Autómato Fatek FBs-10MCR2-AC e expansão FBs-CM55E montados no quadro elétrico

O programa utilizado para programar os autómatos é o *Winproladder* da *Fatek*. Este programa foi instalado no sistema operativo Windows XP e encontra-se representado na Figura 4.9.

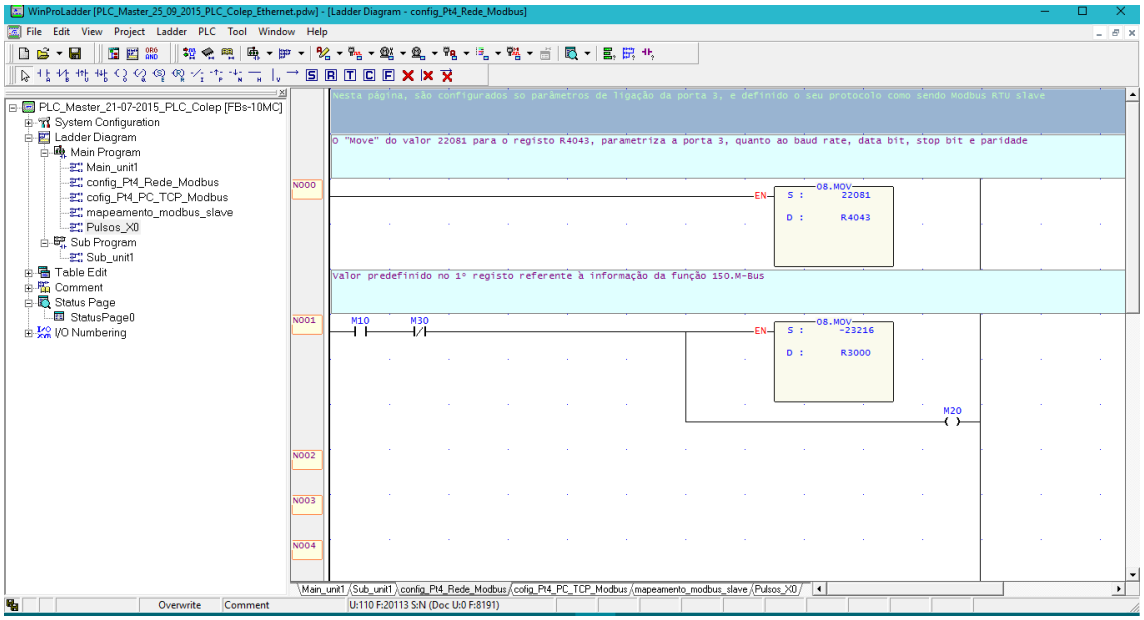


Figura 4.9 – Software Winproladder

Este software permite a programação dos autómatos da *Fatek*. A linguagem de programação é *Ladder*. Trata-se de uma linguagem gráfica de programação de *PLC's*, que tem como principais

símbolos relés, botões, solenoides. Para funções compostas, a representação é feita por caixas com a indicação do tipo de função.

Para transferência do programa do computador para o autómato, foi utilizado um cabo de comunicação série RS232, representado na Figura 4.10, em que uma das pontas do cabo tinha uma interface DB9 e na outra que ligava ao autómato, tinha uma *Mini Din*. Esta interface *Mini Din* foi ligada à porta 0 (*Port0*) do autómato que se encontra do seu topo.

Figura 4.10 – Cabo utilizado para a transferência do programa para o autómato [29]

4.2.7 Programação do autómato para comunicação com o contador inteligente

Com o contador de energia configurado, foi necessário programar e configurar o autómato para este recolher os valores do contador com comunicação inteligente (*Ducati Smart Più*).

A porta utilizada para a comunicação foi a porta 3 da expansão (*Port 3*), pois a porta 4 (*Port 4*) possui duas interfaces, RS485 e Ethernet, mas ambas estão representam a porta 4, ou seja, apenas se pode usar uma das interfaces mediante o tipo de ligação física que é necessária.

Então, esta porta 3 teve de ser configurada de maneira a que os parâmetros da comunicação série fossem os mesmos que foram escolhidos para o contador. Relembrando esses parâmetros, estes eram: 9600 bits/s, 1 stop bit, sem bit paridade, e 8 bits de dados.

Consultado o manual do aparelho capítulo 12 e secção 12.4.3 [30], foi possível verificar que a posição de memória do autómato R4043, é a responsável pela configuração dos parâmetros de comunicação série desta porta. Sendo uma memória do tipo “Special Register” [31], tem capacidade para 1 “word” (16bits = 2 Bytes). Segundo o a informação contida na secção 12.4.3, o byte mais significativo tem sempre o mesmo valor, 56H (sendo H a indicação que o número se encontra em base hexadecimal), enquanto que o byte menos significativo é onde se encontra a informação da parametrização da porta 3.

O byte menos significativo (8 bits) é dividido assim da seguinte maneira:

- 4 bits menos significativos: para a opção de 9600 bits/s o valor no manual é 0001b (sendo b a indicação que o número se encontra em base binária)
- 5º bit menos significativo: para a opção de um 1 stop bit o valor é 0b.

- 6º bit menos significativo: para a opção sem paridade o valor é 0b
- 7º bit menos significativo: a opção de 8 data bits é representada pelo 1b
- 8º bit menos significativo: como definimos sem paridade não interessa o valor deste bit, por isso pode ser colocado a 0b que significa par

Assim, o segundo byte é 01010001. Para converter para hexadecimal, os bits podem ser organizados 4 a 4 e cada conjunto de 4 bits representa um caracter hexadecimal. Ou seja, este número pode ser escrito da seguinte forma: 0001b=1H e 0101b=4H.

Então o byte menos significativo é 41H, sendo o valor a ser colocado na memória R4043 de 5641H. Para colocar este valor na posição de memória, foi utilizada uma função do software *Winproladder*, chamada de MOV com o número 8. Esta função quando lhe é ativado o input *enable*, EN na Figura 4.11, realiza a gravação do valor do parâmetro *source*, S na Figura 4.11, na posição de memória definida como *destination*, D na Figura 4.11. Como a linha está ligada diretamente ao lado esquerdo da imagem, que representa o valor lógico 1, a cada ciclo que o autómato realiza, esta ação é efetuada.

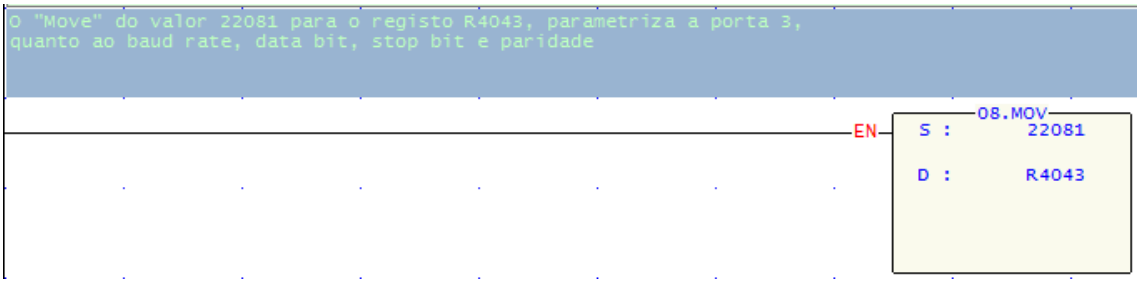


Figura 4.11 – Representação da função 08.MOV

Neste momento a porta 3 encontra-se com os meus parâmetros de configuração que o contador inteligente. Mas agora será necessário o autómato enviar o pedido das posições de memória de interesse, para obter os dados do contador inteligente. Para isso é necessário aplicar o protocolo *Modbus RTU* à porta 3. O *Winproladder* disponibiliza uma função para esse fim. A função 150P.M_BUS, representada na Figura 4.12.

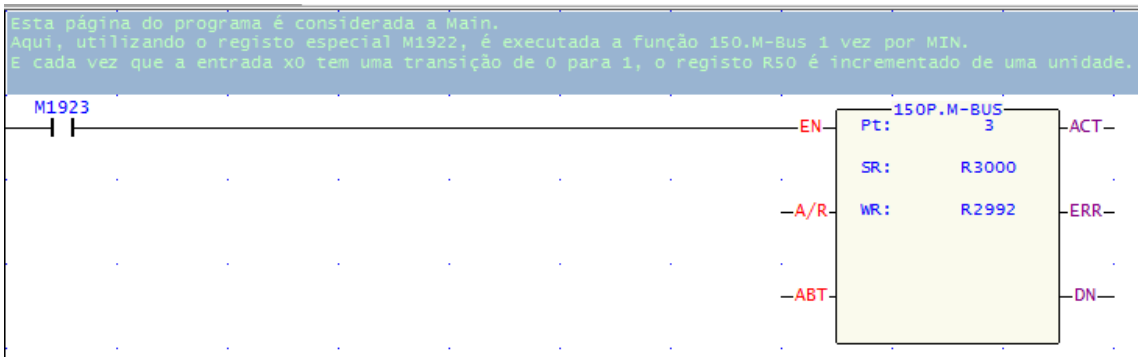


Figura 4.12 – Representação da função 150P.M_BUS

Esta função, só é executada quando o input *enable* tinha o valor lógico um. Isso só acontecia quando o contacto normalmente aberto M1923 também tinha valor lógico 1, o que apenas acontecia apenas uma vez a cada segundo, isto porque esta memória M1923 trata-se de um *clock* de 60s.

Ao utilizar esta função é necessário definir 3 valores. ***Pt*** é a porta que irá transmitir a informação, neste caso é a 3. ***SR*** é o registo inicial onde é colocada a informação que vai conter a mensagem *Modbus*. E ***WR*** é registo inicial onde irá ser escrita a informação de como decorreu a ação (a ser preenchido pelo autómato) [32].

Para este projeto, foram definidos vários intervalos de memórias R do autómato mediante a função. No caso dos valores necessários para configurar o protocolo *Modbus RTU* pela função 150P.M_BUS, foi definido o início na posição R3000. Definindo então o *SR* como sendo *R3000*, tem de se colocar o valor R300**0** no campo **SR** e colocar os seguintes valores nos registos de **R3000** em diante:

- **SR+0=R3000: A550H** (valor predefinido a colocar no 1º registo)
- **SR+1=R3001: 1º BYTE=07H, 2BYTE=nº** de mensagens *Modbus* a enviar.

A partir deste registo, vamos escrever em **7xN registos sendo N o 2byte do registo R3001**. Ou seja, cada mensagem precisa e 7 registos para ser configurada.

- **SR+2=R3002**: nº do *slave*
- **SR+3=R3003**: código do comando a realizar. Atenção! Este código não é o mesmo que é definido pela norma *Modbus RTU*, mas sim pelo *software da Fatek*. Assim sendo, temos os seguintes valores: 1-ler informação do *slave*, 2-escrever múltipla informação no *slave*, 3-escrever num único registo
- **SR+4=R3004**: nº de posições a ler no *slave*
- **SR+5=R3005**: tipo de dados do *master (PLC)* onde irá ser escrita a informação que venha na resposta. O nº associado a cada tipo de dados encontra-se na **página 46** [32]**.**
- **SR+6=R3006**: Nº do registo onde irá ser escrita a informação. Se queremos que escreva a informação no registo R100, então **R3006=100**
- **SR+7=R3007**: tipo de dados do *slave*. O nº correspondente encontra-se também na **página 46 do capítulo 13** [32].
- **SR+8=R3008**: neste registo colocamos o endereço do valor que queremos ler no *slave*.

A seguir na Tabela 4.4, está descrito o exemplo para as duas transações feitas neste teste.

Tabela 4.4 – Tabela de valores para configurar o pedido da posição 106 e 107 do contador inteligente

SR	Registo R	Valor
0	R3000	$A550_{16}$
1	R3001	0701_{16}
2	R3002	0041_{10}
3	R3003	0001_{10}
4	R3004	0002_{10}
5	R3005	0012_{10}
6	R3006	1410_{10}
7	R3007	0004_{10}
8	R3008	106_{10}

Como mais à frente vai ser possível verificar, estes valores são escritos externamente nas posições de memória pela aplicação Visual Basic, baseando-se na informação da base dados. Pois, no caso de ser necessário adicionar contadores inteligentes à rede RS485, teríamos de

voltar a reprogramar o autómato. Desta forma, a adição de contadores inteligentes à rede é feita apenas uma alteração na base de dados. Mais à frente vai ser explicado como.

Para esta independência do programa do autómato do nº de *slaves* e da automatização do processo, foi necessário criar uma regra das posições de memória a atribuir a cada *slave* para poder guardar a informação dos pedidos. Assim, foi criada a seguinte regra:

- Foram reservadas 10 posições de memória do tipo R para cada *slave* (ou seja 10 *words*)
- A posição de memória é conseguida através da seguinte regra: 1000 + 10 x nº *slave*

Ou seja, por exemplo o *slave* nº 1 terá para si reservadas as posições R1010 até R1019. Neste caso em particular, o contador é o *slave* nº 41, ou seja, as posições de memória R1410 até R1419 são-lhe reservadas segundo esta regra que foi criada. Mais à frente no documento, vai ser mais percetível a maneira como foi automatizado este processo de adição de contadores inteligentes à rede RS485.

Quanto aos registos do tipo WR, é onde vai ser escrita a informação de como correu a comunicação. Ocupam 8 registos e a informação detalhada encontra-se na **página 46, capítulo 13 do manual** [32]**.**

4.2.8 Programação do autómato para a recolha dos valores dos contadores de output de impulsos

Para fazer a leitura dos contadores de output de impulso elétrico, foram utilizadas as entradas digitais do autómato. O autómato em causa, *Fatek Fbs-10MCR2-AC*, possui 6 entradas digitais. As 6 entradas dividem-se em categorias: duas entradas de 200KHz de frequência máxima, duas de 20 KHz e duas de 5 KHz. Por maior que seja o consumo que um contador esteja a medir, é impossível o output elétrico atingir estas frequências. Por isso este aspeto da frequência da entrada digital foi desprezado.

No caso particular desta rede de teste, um par de fios, um azul e um castanho, foram ligados ao contador como podemos ver na Figura 4.13, no seu canto inferior esquerdo. O castanho foi ligado no conetor positivo e o castanho no negativo.

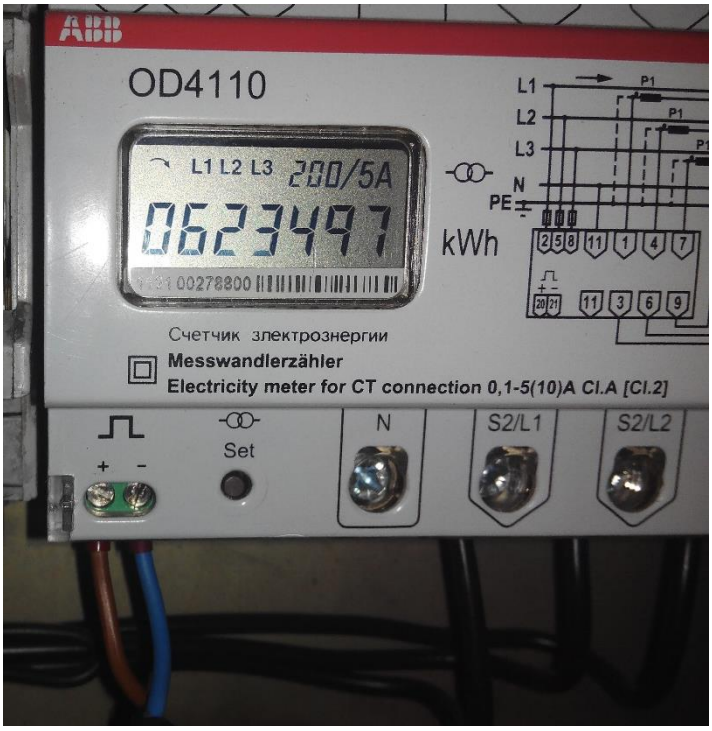


Figura 4.13 – Ligação do output de impulso elétrico do contador da rede de teste

Do lado do autómato, a ligação foi feita da seguinte forma. À entrada X0 do autómato foi ligado o fio neutro, o fio azul. À saída do autómato que fornece 24V DC foi ligado o fio castanho. Relembrando, o contador apenas fecha o contacto interno, não fornecendo qualquer tipo de energia ao circuito. O conector "SS" do autómato não é mais que um contacto comum às entradas do tipo "X", e este foi ligado ao polo menos do 24V DC, para fornecer "neutro" à entrada X0. O esquema representado na Figura 4.14, explica estas ligações descritas.

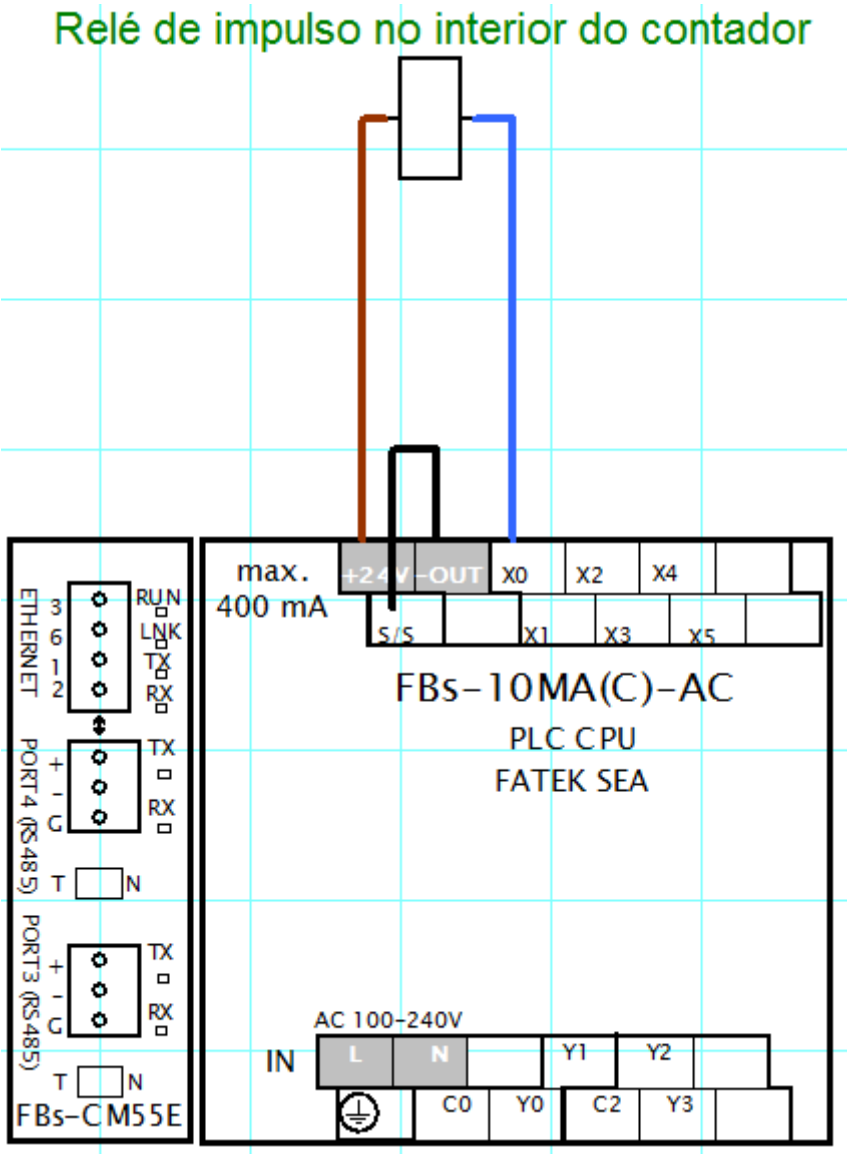


Figura 4.14 – Representação da ligação do autómato ao contador ABB

Com esta ligação, foi possível ativar a entrada X0 a cada impulso do contador. De seguida foi necessário incrementar uma variável do autómato que tomasse sempre o a valor da energia consumida no contador. Foi preciso também ter em conta que o limite do valor de energia consumida deste contador. Ou seja, quando os dígitos do display atingem todos o valor "9", este reinicia a 0.

No entanto, este valor máximo era de 7 dígitos todos com o valor 9 (9999999). No entanto as memórias do autómato ("Registers"), apenas possuíam 16 bits (1word ou 2 bytes), que pode tomar o valor máximo de 2^{16}, que é igual a 65535 (abdicando do sinal) e que é menor que 9999999. Então tiveram de ser utilizadas duas memórias do tipo R. Uma representando os 16 bits menos significativos e outra os 16 bits mais significativos. Assim o número limite passou a ser 2^{32} que é 4294967296. No entanto, o facto de se utilizar 2 posições de memória distintas, dificultou a tarefa de incremento, pois foi necessário incrementar a posição de memória com os 16 bits menos significativos e depois a posição com os 16 bits mais significativos. O código *Ladder* correspondente a esta tarefa encontra-se representado na Figura 4.15.

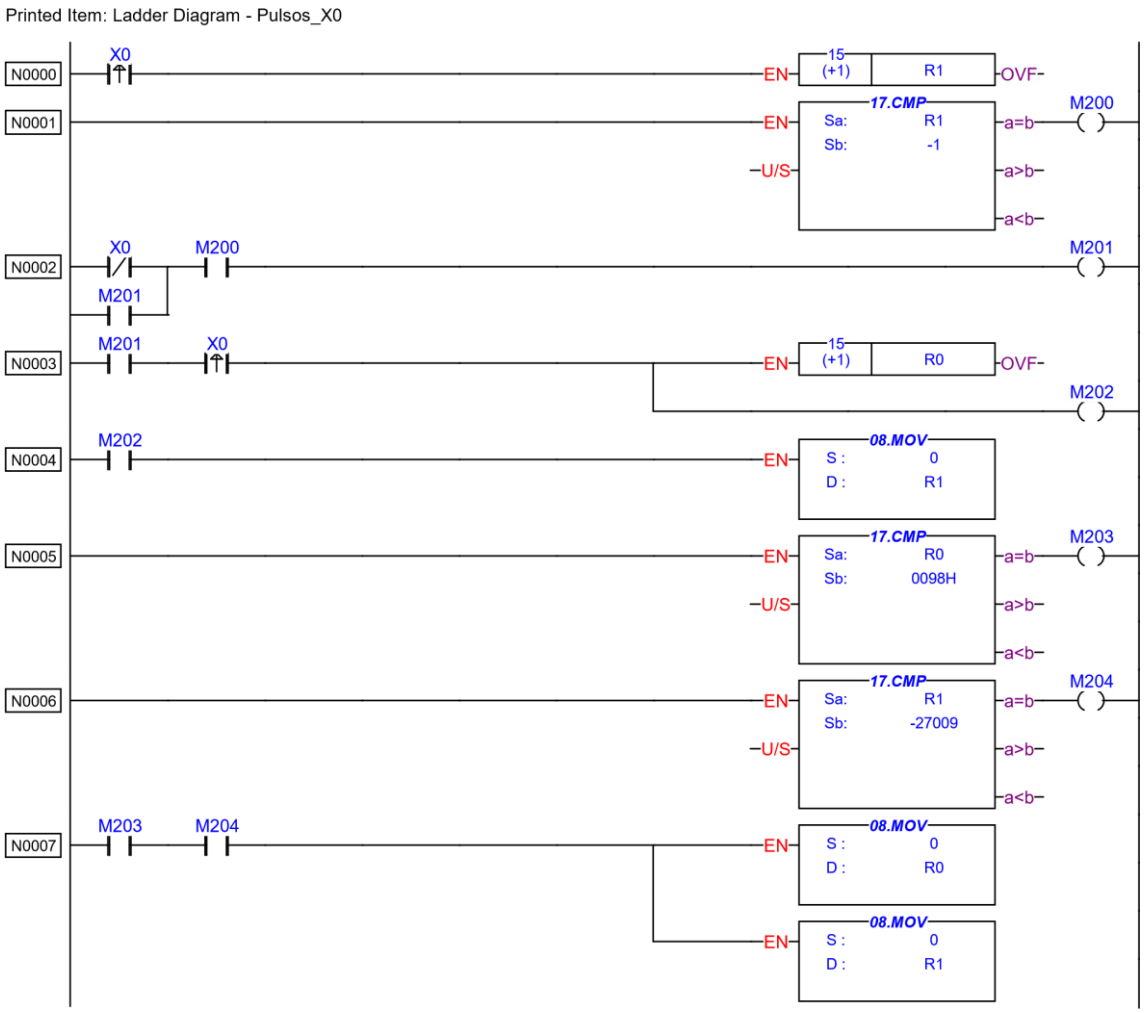


Figura 4.15 – Código Ladder para fazer incremento às duas posições de memória

O limite do contador é 9999999_{10} que é igual a $98967F_{16}$, e este foi dividido em dois *words* (2 bytes) para poder ser guardado nas memórias do tipo R deste autómato. A posição de memória R0 representa o *word* mais significativo, a memória R1 representa o *word* menos significativo.

Assim, para fazer o incremento como se as duas memórias fossem uma só, o incremento foi feito no R1, o *word* menos significativo, e quando este atingia o valor $FFFF_{16}$, no impulso seguinte este valor passava a 0 e a memória R0, *word* mais significativo era incrementado. Este processo repetia-se até, R1 tomar o valor $967F_{16}$ e o R0 tomar o valor 0098_{16}, quando tomavam este valor as memórias eram colocadas a 0.

Neste caso foram reservadas duas posições de memória e também foi aplicada uma regra. Para a entrada X0, foram reservadas as entradas R0 e R1, para X1, R2 e R3 e assim sucessivamente.

4.2.9 Autómato no modo *slave* no *Modbus TCP/IP*

Como já explicado, os valores de interesse para o HMI (computador e base de dados) foram lidos do autómato através de *Modbus TCP/IP*. Para isso foi necessário configurar o autómato como *slave* e de maneira a que este esteja pronto para responder aos pedidos *TCP/IP*.

Por defeito as portas do autómato estão com o protocolo *Fatek*. Para alterar esta definição, é necessário utilizar o registo R4047, como indicado na **secção 12.4.2** [30]. Mediante o valor colocado nesta posição de memória, o protocolo em vigor nas várias portas do autómato e

expansão é alterado. No caso particular da porta 4, o objetivo foi colocar esta porta no modo *slave* do protocolo *Modbus RTU*, e consultando a secção referida anteriormente, um dos valores possíveis foi **R4047=5680$_{16}$**. Este valor coloca todas as portas com o protocolo proprietário da *Fatek*, com exceção da porta 4, que passa a estar no protocolo *Modbus RTU* no modo *slave*.

No entanto existiu um problema. À exceção das funções 5 e 6 do protocolo *ModBus RTU*, o autómato não está preparado para interpretar as restantes funções. Para isso foi preciso fazer um "mapeamento da memória" (*address mapping*). Outro problema é o facto de segundo o protocolo *Modbus RTU*, a referência às posições de memória ser feito através de números, e no caso deste autómato quando nos referimos a uma posição de memória referimos uma letra, que distingue o tipo de memória e o número. Ou seja, se enviamos um pedido ao autómato para ler a posição 30, este não sabe se é R30 ou D30 por exemplo. Assim, o "mapeamento da memória" serviu também para atribuir endereços *Modbus* às posições de memória do autómato com interesse.

Então foi necessário fazer o "mapeamento da memória" para o autómato responder à função 4 do protocolo *Modbus*, "*Read Input Registers*", e para a função 6 "*Preset Single Register*". Para isso foi necessário consultar a página 50 do capítulo 13 [33], e consultando a tabela presente nessa página foram identificadas as posições de memória a alterar.

- R3968=A55AH significa que o utilizador vai utilizar um novo mapeamento dos endereços para a comunicação com o protocolo *ModBus RTU slave*.
- R3975= valor de início do endereço *ModBus RTU* utlizado pelo master. Aplicável à função 4.
- R3976= valor de início do endereço referente à gama de registos do tipo R do autómato que vão ser lidos. Aplicável à função 4.
- R3977= intervalo de registos referidos nos Registos R3975 e R3976 que vão ser lidos.
- R3978= valor de início do endereço *ModBus RTU* utlizado pelo master. Aplicável à função 3,6 e 16.
- R3979= valor de início do endereço referente à gama de registos do tipo R do autómato que vão ser lidos. Aplicável à função 3,6,16.
- R3980= intervalo de registos referidos nos Registos R3975 e R3976 que vão ser lidos.

Então foi feita a seguinte configuração representada na Figura 4.16.

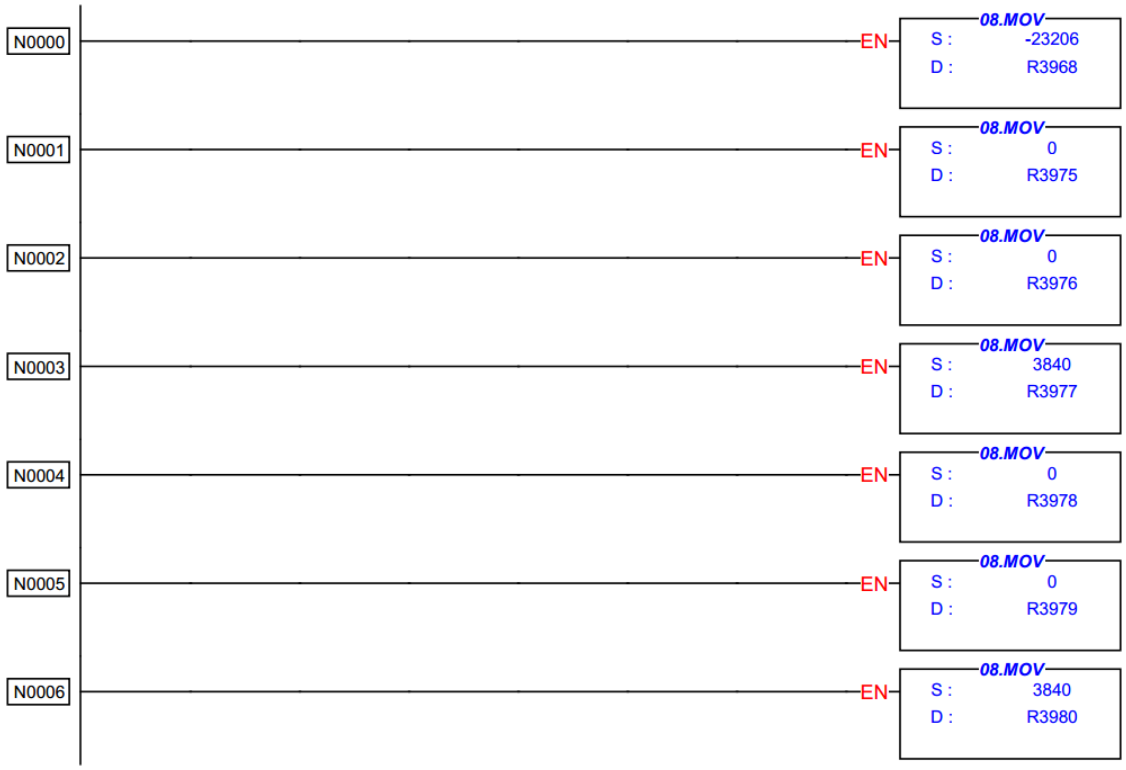


Figura 4.16 – Mapeamento da memória do autómato

Com estes valores quer dizer que passaram a poder ler-se 3840 registos a começar em R0 (R3976 e R3979) até ao registo R3839 (R3977 e R3980), enviando como endereço de memória desde 0 a 3839 (R3975 e R3978). Ou seja, para qualquer uma das funções (3,4,6 e 16), quando o pedido enviado pelo master ao autómato, o endereço de memória a ler é n, vai corresponder a Rn, sendo n um número entre 0 e 3839.

4.2.10 Configuração da ligação TCP/IP da porta 4 do autómato

Para inserir o autómato na rede de internet da empresa, foi necessário configurar a ligação de *Ethernet*, atribuindo-lhe um IP, uma Mascara de Rede, um *Gateway* e um *DNS*. Para isso foram contactados os serviços informáticos da empresa que forneceram os valores presentes na Tabela 4.5:

Tabela 4.5 – Valores de configuração da rede Ethernet do autómato

IP	192.168.19.204
Mascara de Rede	255.255.252.0
Gateway	192.168.16.254
DNS	192.168.16.6

Depois de obtidos os valores foi utilizado um software da *Fatek*, o *Ethernet Module Configuration*, representado na Figura 4.17. Para a configuração do autómato, foi necessária a instalação deste software no computador, e ligar o autómato ao computador, utilizando um cabo de *Ethernet*. Este cabo foi ligado ao autómato na sua porta 4, interface *Ethernet*, e no computador utilizou-se a interface RJ45. Depois da ligação, selecionou-se a opção "Intranet", visível na Figura 4.17.

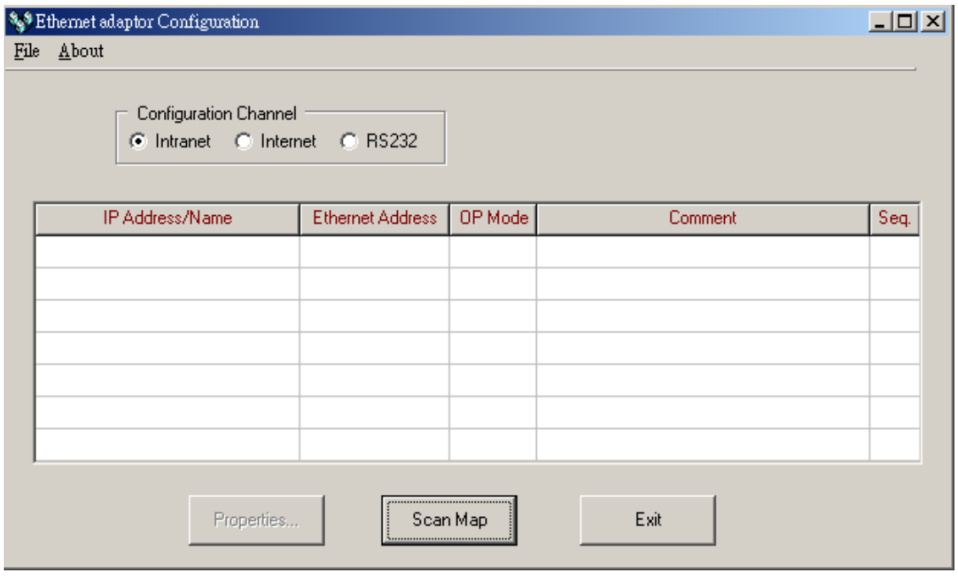


Figura 4.17 - Ethernet Module Configuration

No entanto, depois de selecionar a opção "*Scan Map*", não foi possível encontrar o autómato. Então teve de se utilizar um procedimento complementar. Foram alteradas as propriedades da placa de Ethernet do computador por forma a que o IP desta fosse bastante próximo do IP do autómato. Para isso foi necessário consultar o manual da expansão FBs-CM55E [34], e na página 19, pode se vêr que o IP por defeito desta placa é: 192.168.1.3. Assim foi possível configurar a placa de Ethernet do computador como mostra a Figura 4.18.

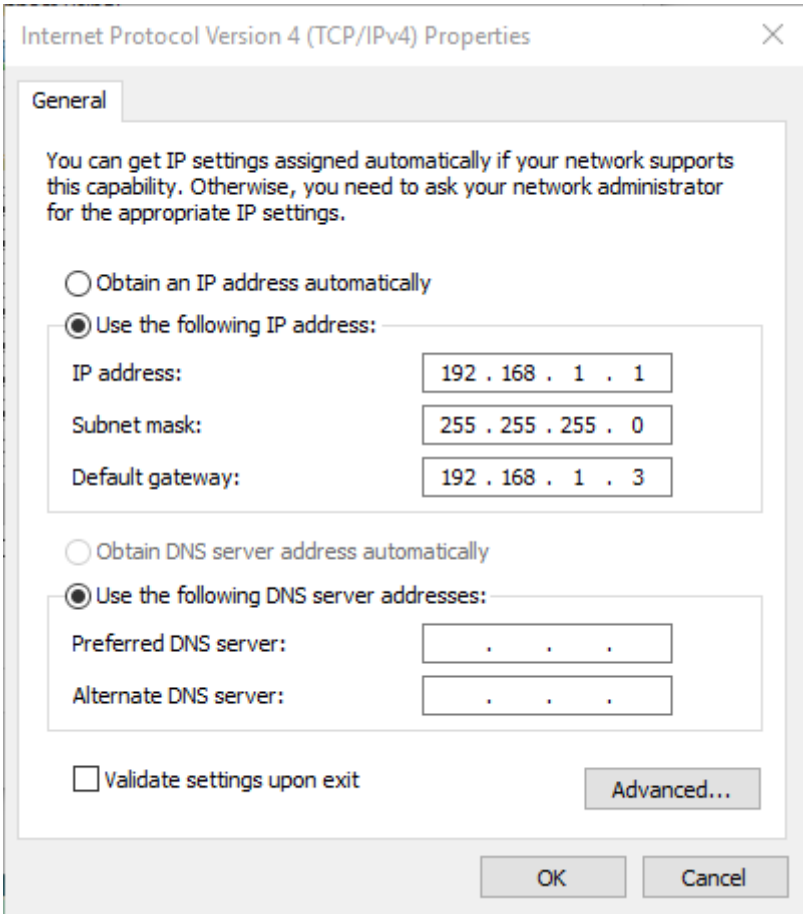


Figura 4.18 – Propriedades da placa Ethernet do computador

Depois desta configuração já foi possível encontrar o autómato com o *software* da Figura 4.17, selecionando o botão das propriedades foi possível alterar para os valores da Tabela 4.5, como representado na Figura 4.19.

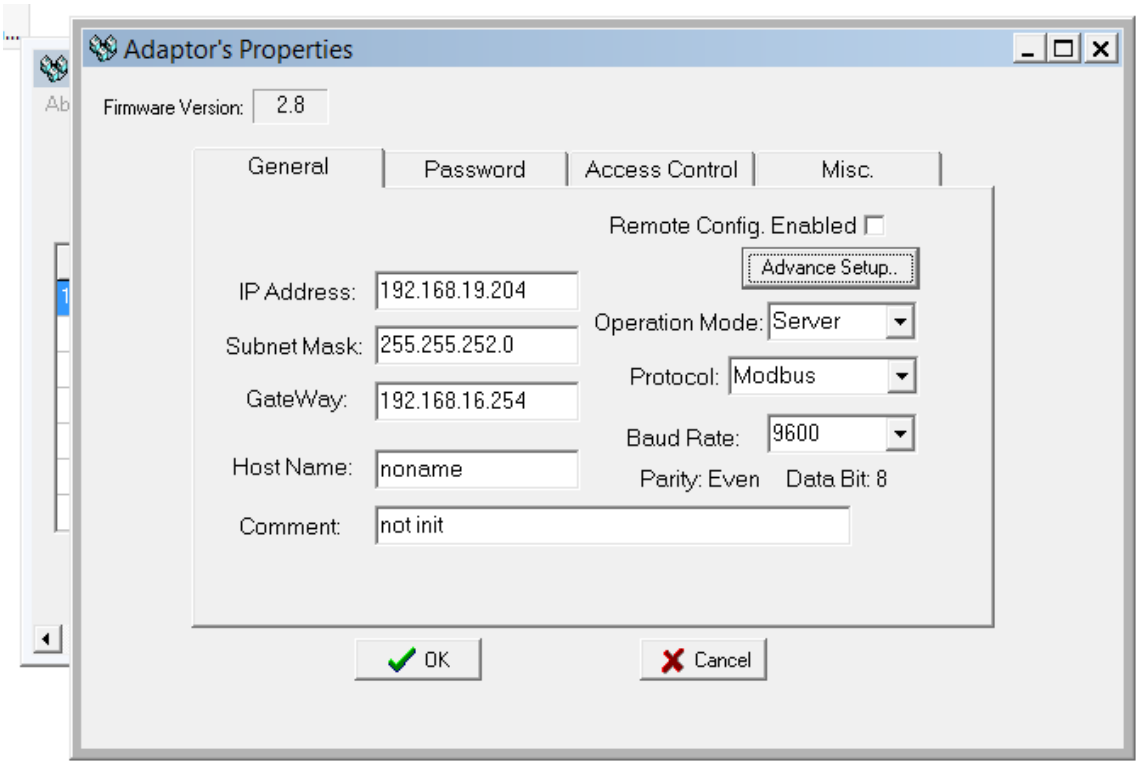


Figura 4.19 – Parametrização da Placa de Ethernet da Expansão do Autómato

Quanto aos parâmetros da ligação, estes não podiam ser alterados à exceção do baud rate. Assim os parâmetros que ficaram definidos para esta ligação foram: paridade par, 8 bits de dados, 9600 bits por segundo.

Desta forma o autómato cumpria todos os objetivos:

- Recolher os valores do contador inteligente
- Recolher os valores do contador com comunicação por impulso elétrico
- Trabalhar como slave no protocolo *Modbus RTU TCP/IP* para o master poder ler os valores das variáveis recolhidas.

4.2.11 Programa em Visual Basic

O programa responsável por fazer a interface do utilizador, desempenhar o papel de master do autómato e fazer o registo dos valores da contagem de energia na Base de dados, foi desenvolvido em Visual Basic, e está representado na Figura 4.20.

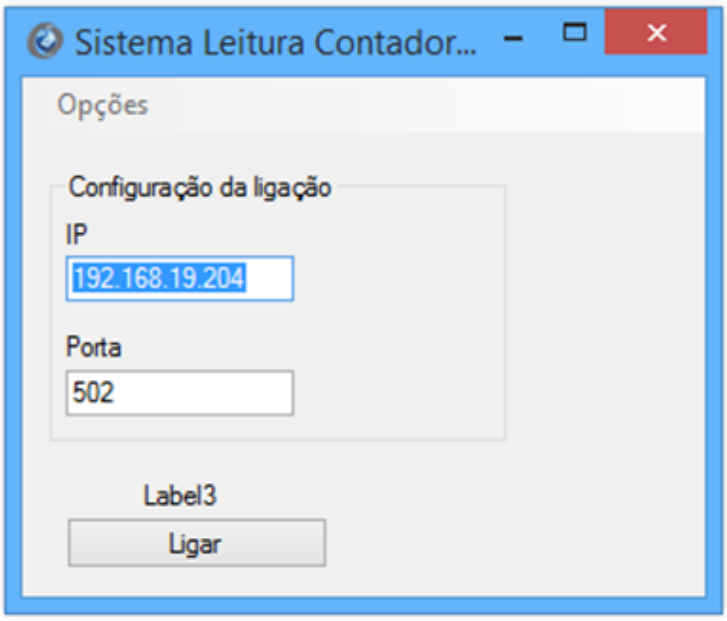


Figura 4.20 – Programa desenvolvido em Visual Basic

Esta interface é a apresentada depois de se iniciar o executável. Ao carregar no botão "Ligar" este programa, através de uma ligação TCP/IP e com parâmetros pré-definidos, tenta a ligação ao aparelho com o IP que está na caixa de texto "IP" através da porta 502 que corresponde ao protocolo Modbus. Depois de a ligação se fazer com sucesso, era possível utilizar o botão "Opções" que permitia realizar uma série de tarefas, como mostra a Figura 4.21, tais como exportar a informação para o Excel, atualizar o valor das posições de memória referentes aos contadores por impulso, alterar o número limite de registos da base de dados e por fim eliminar todos os registos da base de dados.

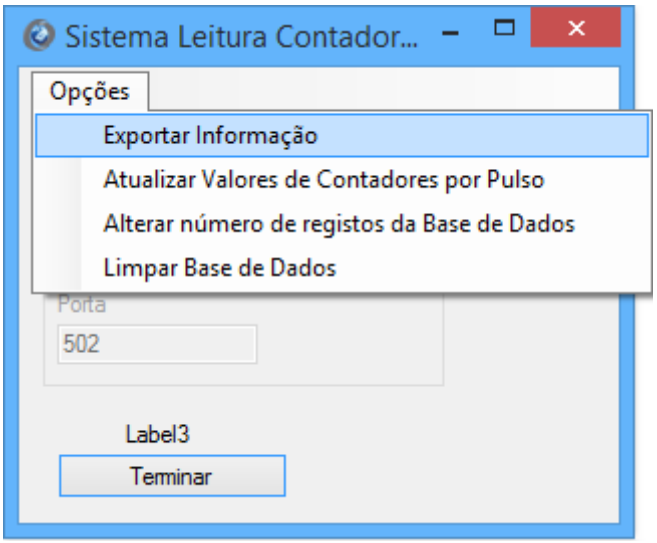


Figura 4.21 – Opções do programa

Para perceber melhor o funcionamento do programa o esquema representado na Figura 4.22, explica o funcionamento deste.

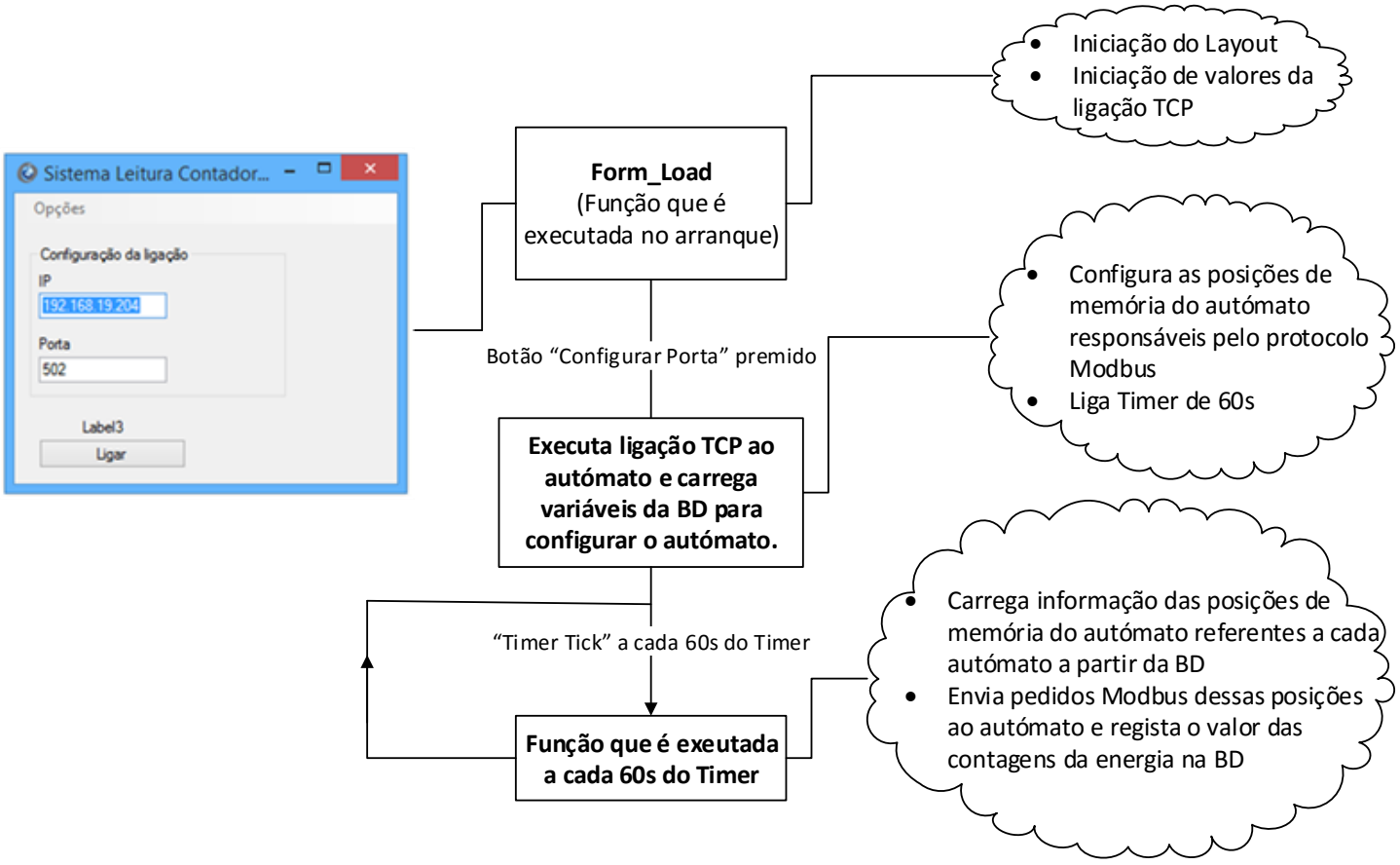


Figura 4.22 – Diagrama do funcionamento do programa

A função "Form_Load" é uma função que é executada quando o programa inicia. Nesta função os valores da ligação TCP/IP, caso do "IP" e "Porta", são inicializados com o valor do IP do autómato e a porta 502.

Quando o botão "Ligar" é premido, é feita a ligação ao autómato. Se esta ligação for feita com sucesso, de seguida é feita a ligação à base de dados. Caso alguma destas ligações não seja possível, é mostrada uma mensagem ao utilizador. No caso das ligações se realizarem com sucesso, é chamada uma função que realiza a configuração do protocolo Modbus RTU que o autómato utiliza para comunicar com o contador de energia inteligente. Esta configuração não é nada mais do que a colocação de certos valores nos registos de memória do autómato que configuram a função 150P.M_BUS como já foi visto anteriormente. Os valores a colocar nestes registos estão presentes numa tabela presente na base de dados. Estes valores contêm uma lista dos contadores existentes na rede Modbus RTU RS485, os seus endereços slaves, e as posições de memória que têm de ser lidas. Esta função é extremamente importante pois permite a adição de contadores inteligentes à rede RS485 Modbus RTU sem ter de se reprogramar o autómato, ou o programa VB.net. Desta forma é garantida a escalabilidade da rede de contadores. No caso dos contadores de impulso elétrico, o mesmo foi feito. Existe uma tabela em que é indicada a entrada do autómato a que está ligada e o nome do contador, e desta forma, o programa respeitando as regras de atribuição de endereços do autómato referidas na secção 4.2.7 e secção 4.2.8.

Por último é ativado um Timer de 60 segundos. Cada vez que passam os 60 segundos, uma função é executada. Esta função é onde é feita a leitura dos valores do contador e seguidamente é feito o registo dos valores na base de dados.

Para executar os pedidos Modbus TCP/IP para o autómato foi pensada uma solução, uma leitura assíncrona. Às vezes as funções podem levar muito tempo a retornar um valor ou executar uma tarefa. Neste caso, se por alguma razão a comunicação TCP/IP demore por alguma razão, ou o número de pedidos aumente, esta tarefa pode atrasar o programa. Assim a solução assíncrona foi a melhor solução encontrada. Foi utilizado um método de seu nome "BeginRead" e utilizada uma função como "Callback" para este método. Nesta função a informação do buffer era recolhida e colocada numa variável global. Desta forma foi realizada a leitura através da ligação TCP/IP.

Quanto a outras opções presentes neste programa, o utilizador podia exportar o histórico de leituras entre duas datas, escolher um período de amostragem inferior a 60 segundos como mostra a Figura 4.23, e esta informação iria ser exportada para uma folha de Excel.

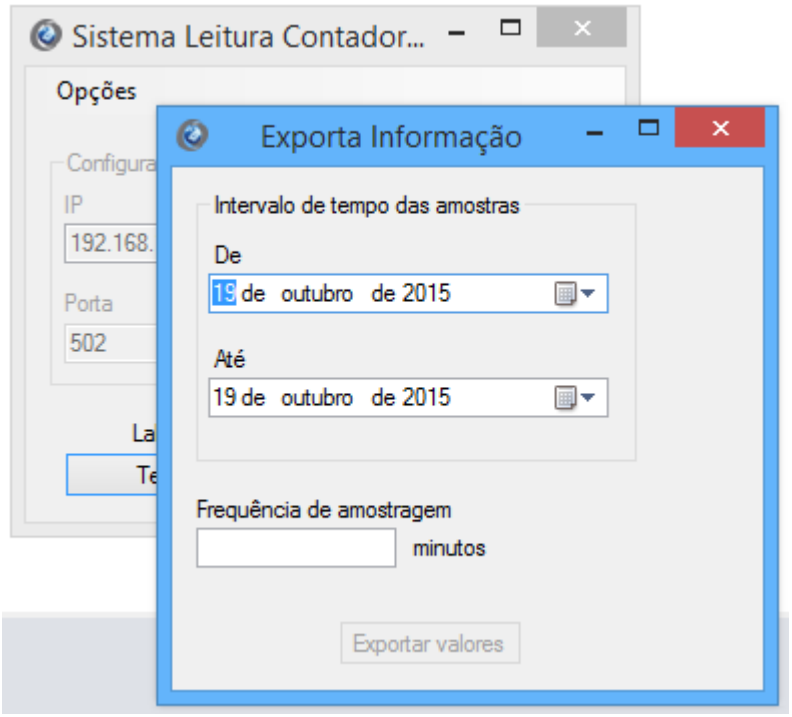


Figura 4.23 – Opção de exportar informação para uma folha de Excel

Outra opção que o programa tinha era a atualização dos valores dos contadores com saída de impulso elétrico como mostra a Figura 4.24.

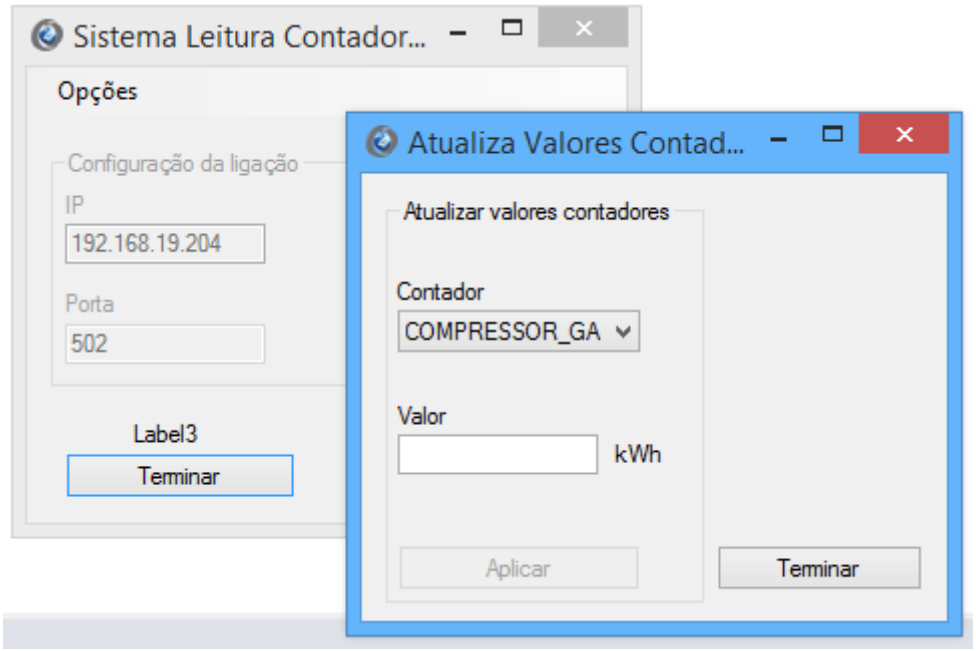


Figura 4.24 – Opção do programa de atualização dos valores dos contadores com saída de impulso

O autómato neste caso apenas incrementava um valor que tinha de ser inicializado pelo utilizador. Esta ação tinha de fazer de maneira síncrona por forma a não ficar desfasado com o valor real do contador.

Ainda outra operação possível de realizar era alterar o número limite de registos na base de dados, como representado na Figura 4.25 para que esta não aumentasse indefinidamente.

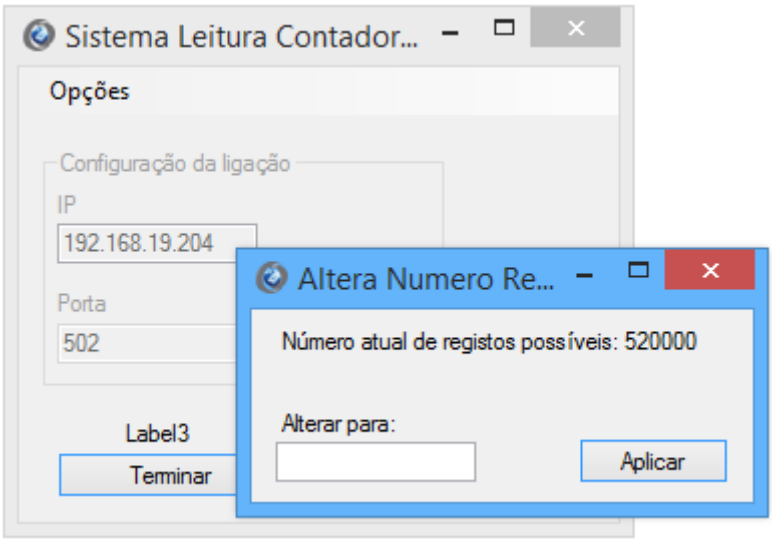


Figura 4.25 – Opção do programa do número de registos limite da base de dados

A última opção disponível era apagar todos os registos na base de dados. Obviamente todas estas opções de alterações críticas teriam de ser confirmadas em "Message Box" de aviso, a perguntar se o utilizador estava seguro da decisão.

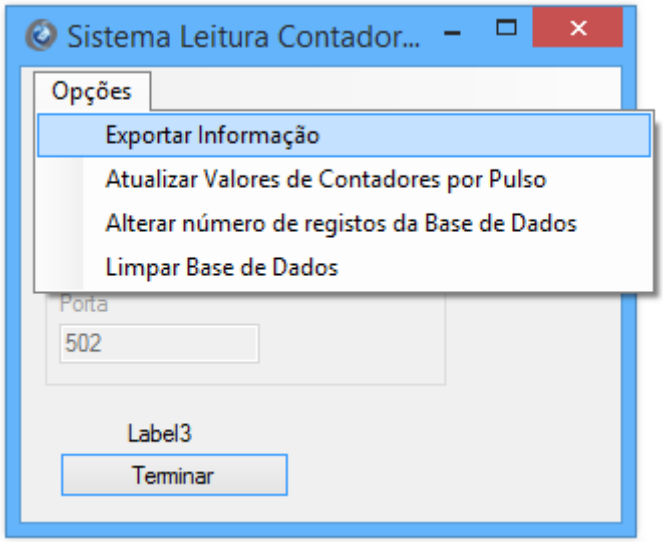


Figura 4.26 – Opção de eliminar os registos todos da base de dados

4.2.12 Base de dados

A base de dados utilizada foi o Microsoft Access, uma ferramenta do Microsoft Office. Esta base de dados tinha como função servir para o registo das contagens de energia dos dois contadores desta rede. Outra utilidade que teve foi servir de registo dos contadores e da sua informação necessária para a configuração do autómato quanto aos pedidos *Modbus RTU* que este fazia aos contadores na rede RS485. O tipo de ligação escolhida foi uma ligação ODBC. Este tipo de ligação é configurada no próprio computador Windows e tem como desvantagem que a base de dados se encontre no mesmo computador que a aplicação. No entanto tendo em conta que a configuração da ligação é muito simples e neste caso em particular a base de dados estava destinada a ficar no mesmo computador que a aplicação Visual Basic, então esta foi a melhor solução. Quanto à comunicação entre a aplicação e a base de dados estas foi feita através da linguagem SQL (Structured Query Language).

Como mostra a Figura 4.27, esta base de dados continha três tabelas e uma consulta.

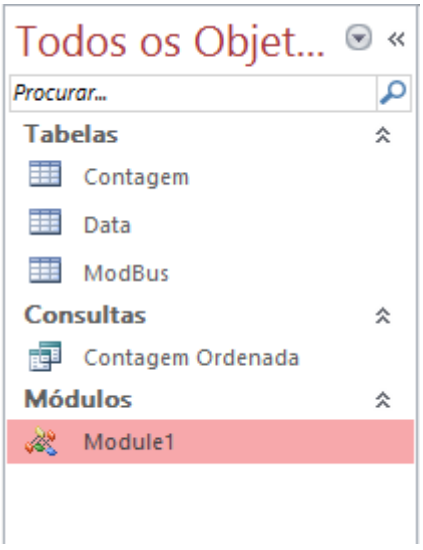


Figura 4.27 – Tabelas e Consultas da base e dados

As tabelas são onde se encontram os registos da base de dados. As consultas são acessos às tabelas que permitem organizar a informação por uma ordem desejada. Esta consulta foi usada para aceder à tabela Contagem de forma a que os registos estivessem organizados de forma cronológica.

A tabela contagem, continha os valores das leituras dos contadores como mostra a Figura 4.28. Os valores apresentados nas colunas dos contadores encontram-se em Wh.

Contagem

data	COMPRESSOR_GA_90_VSD	LINHA_34
01/10/2015 05:33:26	627000900	0
01/10/2015 05:34:26	627002100	0
01/10/2015 05:35:26	627003300	0
01/10/2015 05:36:26	627004500	0
01/10/2015 05:37:26	627006100	0
01/10/2015 05:38:26	627007900	0
01/10/2015 05:39:26	627009300	0
01/10/2015 05:40:26	627010800	0
01/10/2015 05:41:26	627012200	0
01/10/2015 05:42:26	627013700	0
01/10/2015 05:43:26	627015300	0
01/10/2015 05:44:26	627016800	0
01/10/2015 05:45:26	627018200	0
01/10/2015 05:46:26	627019600	50773160
01/10/2015 05:47:26	627020900	50773160
01/10/2015 05:48:26	627022400	50773160
01/10/2015 05:49:26	627023900	50773160
01/10/2015 05:50:26	627025400	50773160
01/10/2015 05:51:26	627026700	50773160
01/10/2015 05:52:26	627027900	50773160
01/10/2015 05:53:26	627029000	50773180
01/10/2015 05:54:26	627030100	50773180
01/10/2015 05:55:26	627031200	50773180
01/10/2015 05:56:26	627032300	50773180
01/10/2015 05:57:26	627033300	50773200
01/10/2015 05:58:26	627034300	50773200
01/10/2015 05:59:26	627035300	50773220
01/10/2015 06:00:26	627036200	50773220
01/10/2015 06:01:26	627037200	50773240
01/10/2015 06:02:26	627038200	50773240
01/10/2015 06:03:26	627039300	50773260

Figura 4.28 – Tabela dos valores da contagem de energia dos dois contadores da rede

A tabela Data referia-se à informação relativa aos contadores com saída de impulso elétrico. A mostra Figura 4.29 a tabela. A coluna “entrada X-” define em que entrada do autómato o contador está ligado. A coluna contador tem o nome do contador que é igual ao que aparece na tabela “Contagem”. A coluna “ID_PLC” refere-se a que autómato o contador está ligado. Isto

acontece, pois, este programa foi desenvolvido a pensar na arquitetura concebida em que a fábrica possui vários computadores locais. A partir deste "ID_PLC" e caso não fosse 0, o programa configuraria a comunicação do autómato master aos outros autómatos para que este lê-se a posição de memória referente no autómato como se de uma posição de memória de um dos contadores inteligentes se tratasse. A coluna "impulsos/kWh" continha o valor de impulsos necessários até ser contabilizado outro kWh de energia pelo contador. Por fim a coluna "razão em W" era o número que era necessário multiplicar para que o registo na tabela "Contagem" tivesse a unidade Wh.

Contagem | Data | ModBus

entrada X-	contador	ID_PLC	impulsos/kWh	razão em W
0	COMPRESSOR_GA_90_VSD	0	10	100
0		0	0	0

Figura 4.29 – Tabela Data da Base de Dados

A tabela *Modbus* refere-se à informação dos contadores com comunicação inteligente. Com esta informação as posições de memória do autómato que configuram a função responsável por enviar os pedidos *Modbus* para a rede de autómatos com comunicação inteligente. A coluna "ID" refere-se ao endereço do autómato, neste caso é 0, a coluna "contador_grandeza" contém o nome do contador que aparece na tabela "Contagem". A coluna "Nº Slave" contém o número do *slave* correspondente ao contador em causa. A coluna "Func" corresponde à função a realizar pelo autómato. A coluna "Nº Pos" é o número de posições a ler. A coluna "Tipo Reg" significa o tipo de registo em que a informação vai ser guardada no autómato. A coluna "Nº Reg" é o número do registo em que a informação vai ser guardada. Este valor obedece à regra estabelecida que foi explicada nas secções anteriores. A coluna "Tipo Data Slave" diz respeito ao tipo de variável a ser lido no *slave*. A coluna "End Modbus" é o endereço a ser lido no contador. A coluna "Relação relativamente a wh" é a multiplicação necessária a fazer para o valor obtido da leitura ser em Wh.

ID	contador_grandeza	Nº Slave	Func	Nº Pos	Tipo Reg	Nº Reg	Tipo Data Slave	End Modbus	Relação relativamente a wh
0	LINHA_34	41	1	2	12	1410	4	106	10
0		0	0	0	0	0	0	0	0

Figura 4.30 – Tabela Modbus da Base de Dados

Desta forma foi explicada toda a solução que foi implementada como rede de teste na empresa para estes dois contadores, que tornou possível obter as leituras dos contadores tal como desejado pelo Energy Maneger. Depois de validada a arquitetura foi necessário orçamentar a implementação deste sistema para toda a fábrica.

4.3 Orçamento para ampliação da rede de teste a toda a fábrica

Validada a arquitetura foi feito um orçamento para implementação do sistema a todos os contadores de energia elétrica da empresa. Assim com o levantamento e mapeamento de todos os contadores de energia elétrica anteriormente, foi possível determinar os autómatos e respetivas expansões necessárias. A quantidade de aparelhos e respetivos preços encontram-se na Tabela 4.6.

Tabela 4.6 – Autómatos necessários à expansão da rede a todos os contadores de energia elétrica

Autómato/Expansão	Quantidade	Preço unitário	Preço total
Fatek FBs-14MAR2-AC	2	147.40 €	294.80 €
Fatek FBs-20MAR2-AC	2	184.26 €	368.52 €
Fatek FBs-32MAR2-AC	1	231.64 €	231.64 €
Fatek FBs-CB55	5	42.12 €	210.60 €

Todo este equipamento perfaz um investimento de 1105,56€ que a somar aos 323,76€ gastos na rede de teste totaliza o valor de 1429,32€.

Para ligar os contadores aos autómatos, tanto os contadores inteligentes como os contadores de saída de impulso elétrico, são necessários dois tipos de cabo. O cabo para a transmissão de dados segundo o protocolo *Modbus RTU* será um cabo com dois fios e malha para evitar ruído na transmissão e possíveis erros de transmissão. O cabo para os impulsos elétricos dos contadores que não possuem comunicação inteligente será um cabo constituído por dois fios apenas. Esses cabos de ligação, o seu preço por unidade de comprimento, a distância necessária e o preço total, encontram-se na Tabela 4.7.

Tabela 4.7 - Cabos necessários

Tipo de Cabo	Quantidade	Preço unitário	Preço total
Cabo com malha (RS485)	1,111 m	2.00 €/m	2,222.00 €
Cabo normal (impulso 24V)	2,815 m	1.00 €/m	2,815.00 €

O dinheiro necessário para adquirir os cabos totaliza um valor de 5037€.

4.3.1.1 Substituição dos contadores sem comunicação e respetivos TI's:

Com o levantamento dos contadores anteriormente feito, foi possível perceber treze dos contadores não possuíam qualquer tipo de comunicação. Neste orçamento realizado, foi comtemplada a sua substituição. Foi decidido que estes iriam ser substituídos por contadores com comunicação inteligente. O modelo escolhido foi o *Seneca S504C-6-MOD-MID*. Mas não basta apenas substituir os contadores de energia elétrica. É necessário também adquirir transformadores de intensidade. Estes transformadores são necessários um por cada fase (neste caso todos os contadores substituídos iriam analisar três fazes), e o seu diâmetro depende da secção do cabo. Assim foi determinado que iriam ser necessários 21 *Seneca FRAT10110005* (d=101 mm) e *18 FRAC5110005* (d=51 mm). Todos estes materiais e o seu preço encontram-se descritos na Tabela 4.8.

Tabela 4.8 - Transformadores de Intensidade necessários

Transformador de Intensidade	Quantidade	Preço unitário	Preço total
Seneca S504C-6-MOD-MID	13	195.48 €	2,541.24 €
FRAC5110005	18	25.45 €	458.10 €
FRAT10110005	21	47.93 €	1,006.53 €

O investimento necessário para a aquisição deste material é cerca de 4005,87€.

Por fim, a somar a todo este investimento foi estimado um custo de mão de obra de 2500€. Assim sendo, o custo total da implementação deste sistema de monitorização a todos os contadores de energia elétrica é de cerca de 12972,19€.

5

Resultados e Conclusões

5 Resultados e Conclusões

Como referido na secção anterior o investimento total deste projeto é de 12972,19€. O consumo anual de energia elétrica na Colep é de cerca de 19656 MWh que equivale a 4226 tep. Segundo um manual elaborado por especialista na área da energia [7] um sistema deste género aplicado à energia elétrica pode poupar entre 2 a 8%. Tomando o valor intermédio de 5%, podem ser poupados cerca de 983 MWh/ano o que equivale a uma redução de cerca de 462 tCO_2/ano. Esta poupança corresponde a 86565€ por ano, o que faz com que o tempo de *payback* seja bastante baixo.

No entanto segundo o SGCIE instalações com consumo anual igual ou superior a 1000tep beneficiam de um ressarcimento de 25% dos investimentos realizados em equipamentos e sistemas de gestão e monitorização dos consumos de energia até ao limite de 10000€. Assim, tirando partido da legislação, esta ajuda reduz o investimento para 9729,14€.

Conclui-se com este trabalho que é possível desenvolver um sistema de monitorização de energia de forma relativamente simples. Quando comparado com a solução *SCADA*, em que as licenças podem atingir valores muito elevados, a solução apresentada neste trabalho é muito interessante do ponto de vista económico e não executando as tarefas a que se propõe de forma inferior. No entanto os sistemas *SCADA* têm a vantagem de possuir ferramentas visuais muito interessantes que podem ajudar bastante o gestor de energia a tomar decisões e a perceber o modo como é gasta a energia. Este tipo de ferramentas visuais ou mesmo de algoritmos de tratamento de dados dos consumos de energia, podem ser bastantes complicados de programar para poder integrar num Software deste tipo como o que foi desenvolvido neste trabalho. Assim o software *SCADA* toma vantagem. Outro aspeto positivo é o facto deste trabalho não ter utilizado qualquer tipo de OPC ou facilidade de comunicação de alguma marca, tanto na ligação aos autómatos como nos contadores. Isto permite que este sistema possa ser atualizado/alterado recorrendo a qualquer marca de autómato ou mesmo um RTU, desde que este comunique segundo os protocolos standard que foram utilizados.

Por último é importante referir que este trabalho visou apenas os contadores de energia elétrica, mas devido à sua arquitetura este projeto pode ser aplicado a qualquer contador/analisador dos mais variados tipos de energia.

6 Referências

[1] APREN, “ESTRATÉGIA NACIONAL PARA A ENERGIA 2020,” 2015. [Online]. Available: http://www.apren.pt/pt/dadostecnicos/index.php?id=206&cat=197. [Accessed: 02-Nov-2015].

[2] Adene, “PNAEE.” [Online]. Available: http://www.adene.pt/programa/pnaee-2016-plano-nacional-de-acao-para-eficiencia-energetica-2016. [Accessed: 03-Nov-2015].

[3] Adene, “Consumo de energia final por setor de atividade.” [Online]. Available: http://www.adene.pt/indicador/consumo-de-energia-final-por-setor-de-atividade. [Accessed: 03-Nov-2015].

[4] Adene, “CONSUMO ENERGÉTICO NA INDÚSTRIA,” 2015. [Online]. Available: http://www.adene.pt/consumo-energetico-na-industria. [Accessed: 02-Nov-2015].

[5] Adene, “SGCIE – METAS.” [Online]. Available: http://sgcie.publico.adene.pt/SGCIE/Paginas/Metas.aspx. [Accessed: 02-Nov-2015].

[6] Adene, “SGCIE - INCENTIVOS.” [Online]. Available: http://sgcie.publico.adene.pt/SGCIE/Paginas/Incentivos.aspx. [Accessed: 02-Nov-2015].

[7] V. Magueijo, M. C. Fernandes, H. a. Matos, C. P. Nunes, J. P. Calau, J. Carneiro, and F. Oliveira, “Medidas de Eficiência Energética Aplicavéis à Indústria Portuguesa: Um enquadramento tecnológico sucinto - PublicaçãoMedidasEficiênciaEnergéticaIndústria-SGCIE.pdf,” p. 109, 2010.

[8] Jornal Oficial da União Europeia, “DIRETIVA 2012/27/UE DO PARLAMENTO EUROPEU E DO CONSELHO de 25 de outubro de 2012,” 2012. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2012:315:0001:0056:PT:PDF. [Accessed: 15-May-2016].

[9] N. 73 Diário da República: I série, “Resolução do Conselho de Ministros n.º 29/2010 de 15 de abril,” 2010. [Online]. Available: https://dre.pt/application/file/131829.

[10] Diário da República: I série No 70, “Resolução do Conselho de Ministros 20/2013, de 10 de Abril,” 2013. [Online]. Available: https://dre.tretas.org/dre/308281/. [Accessed: 24-May-2016].

[11] Adene, “ENQUADRAMENTO E OBJETIVOS.” [Online]. Available: http://sgcie.publico.adene.pt/SGCIE/Paginas/Enquadramento.aspx. [Accessed: 28-May-2016].

[12] Adene, “ÂMBITO DE APLICAÇÃO.” [Online]. Available: http://sgcie.publico.adene.pt/SGCIE/Paginas/ambito-de-aplicacao.aspx. [Accessed: 30-May-2016].

[13] Adene, “SGCIE - PENALIDADES.” [Online]. Available: http://sgcie.publico.adene.pt/SGCIE/Paginas/Penalidades.aspx. [Accessed: 01-Jun-2015].

[14] C. Gould, “What is SCADA? SCADA systems are the backbone of modern industry,” 2015. [Online]. Available: https://inductiveautomation.com/what-is-scada. [Accessed: 02-Nov-2015].

[15] “SIMATIC WinCC Open Architecture Add-ons.” [Online]. Available: https://mall.industry.siemens.com/collaterals/files/33/JPG/G_ST80_EN_00480j.JPG. [Accessed: 01-Jul-2015].

[16] progea, “MOVICON 11 SCADA/HMI.” [Online]. Available: http://www.progea.com/en-us/products/scadahmimovicon11.aspx. [Accessed: 29-Feb-2016].

[17] “Power Consumption Analysis and Energy Efficiency.” [Online]. Available: http://www.progea.com/portals/0/1productNEXT/new-17 - Copy.bmp. [Accessed: 01-Jan-2015].

[18] Wikipédia, “Remote terminal unit - Wikipedia, the free encyclopedia.” [Online]. Available: https://en.wikipedia.org/wiki/Remote_terminal_unit. [Accessed: 04-Mar-2016].

[19] “circutor.” [Online]. Available: http://circutor.com/images/imatges/documentacion/articulos/FN-cvm-net-4-plus-ahorra-costes.jpg. [Accessed: 01-Jul-2015].

[20] F. D. Petruzella, *Programmable Logic Controllers*, 3rd ed. Mc Graw Hill, 2005.

[21] I. Motorola, “SCADA Systems,” 2007. [Online]. Available: https://www.motorolasolutions.com/content/dam/msi/Products/scada-systems/SCADA_Sys_Wht_Ppr-2a_New.pdf.

[22] Z. Lin and S. Pearson, “An inside look at industrial Ethernet communication protocols,” 2013.

[23] Modbus Organization, “Modbus FAQ.” [Online]. Available: http://www.modbus.org/faq.php. [Accessed: 01-Apr-2015].

[24] RTA automation, “MODBUS TCP/IP.” [Online]. Available: http://www.rtaautomation.com/technologies/modbus-tcpip/. [Accessed: 04-Apr-2016].

[25] ABB Automation Technology Products AB, “ODIN Meter An electricity energy meter from ABB.” [Online]. Available: http://www.visility.com/media/2918/abbbimaaler.pdf.

[26] Ducati, “SMART Più - Technical Characteristics.” p. 2.

[27] Ducati, “MODBUS-RTU Protocol (Mach SMART / SMART Più / DUCA47(-72)-SP / DUCALCD / DUCA-LCD96).” .

[28] Ducati, “SMART Più USER’S MANUAL.” .

[29] Aliexpress, “Rs422 DB9P para 4 P conector Mini Din PLC cabo para Fatek ces / EasyView MT6000.” [Online]. Available: https://www.google.pt/search?q=Fatek+cable+port+0&espv=2&biw=1366&bih=643&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiYwsfP5arNAhXGAxoKHaSvCEgQ_AUIBigB#tbm=isch&q=Fatek+cable+port0&imgrc=EQsnWdXUrHdO8M%3A. [Accessed: 16-Jun-2016].

[30] T. Fbs-plc, S. Interface, and M. Rtu, “Chapter 12 The Communication Function of FBs-PLC,” pp. 1–44.

[31] Fatek Automation Corp, “Chapter 1 PLC Ladder Diagram and the Coding Rules of Mnemonic,” no. 1. p. 15.

[32] T. Methods and N. C. Positioning, “Chapter 13 The NC positioning control of FBs-PLC,” pp. 1–33.

[33] H. S. Link, “Chapter 13 The Applications of FBs-PLC Communication Link Application for FUN151 Instruction,” pp. 1–53.

[34] Fatek Automation Corp, “Ethernet Module User ’ s Manual.” p. 22, 2004.

[35] Rohtek Automation, “FBS-20MCR2-AC (D24), FATEK PLC.” [Online]. Available: http://rohtek.com/p_shop/product.php?id_product=35. [Accessed: 04-Jun-2015].

[36] Fatek, “FBs-CM25E.” [Online]. Available: http://www.fatek.com/en/prod.php?act=view&no=52. [Accessed: 04-Jun-2015].

[37] Janitza, “Janitza-Manual-UMG604-en.pdf.” [Online]. Available: http://www.janitza.com/manuals-current-devices.html. [Accessed: 16-Apr-2015].

[38] Fatek, “Advanced Function Chapter 13 : The Applications of FBs-PLC Communication Link.” [Online]. Available: http://www.fatek.com/en/data%2Fftp%2FPLC%2FFBs_Manual%2FManual_2%2FChapter_13.pdf. [Accessed: 04-Jun-2015].

[39] Chaos Computers, “Asus X555LN 15.6” Core i7 Laptop With Dedicated Graphics.” [Online]. Available: http://www.chaosclaremont.co.za/product/asus-x555ln-15-6-i7-4510u/. [Accessed: 16-Apr-2015].

[40] Ebay, “USB to RS485 USB-485 Converter Adapter Support Win7 XP Vista Linux Mac OS WinCE5.” [Online]. Available: http://www.ebay.co.uk/itm/USB-to-RS485-USB-485-Converter-Adapter-Support-Win7-XP-Vista-Linux-Mac-OS-WinCE5-/171511624608. [Accessed: 16-Apr-2015].

[41] Chipdip, “UMG 104 ,(код 52.20.001) универсальное изм. устройство.” [Online]. Available: http://www.chipdip.ru/product1/8748445495/. [Accessed: 16-Apr-2015].

[42] Wikipédia, “Blue P+N+E, 6h.” [Online]. Available: http://en.wikipedia.org/wiki/IEC_60309. [Accessed: 16-Apr-2015].

[43] Los Chispas De Virgen, “Conexion de las bobinas de un motor trifasico.” [Online]. Available: http://red.gnoss.com/pt/comunidade/LCDV/recurso/conexion-de-las-bobinas-de-un-motor-trifasico/d05d793b-939c-4dc7-97e2-f8b77317f55c. [Accessed: 16-Apr-2015].

[44] Wikipédia, “SQL.” [Online]. Available: https://pt.wikipedia.org/wiki/SQL. [Accessed: 04-Aug-2015].

Anexo A

A.1 Planta da Empresa

Anexo B

B.1 Lista de contadores de energia elétrica

Tabela B.1 – Lista de contadores da energia elétrica

LEGENDA NA PLANTA	MARCA	MODELO	COMUNICAÇÃO	Zona
1	MERLIN GERIN	ME4 ZRT	PULSE	3
2	REGULADORA	-	0	3
3	MERLIN GERIN	CER	PULSE	3
4	MERLIN GERIN	PM 700	0	3
5	MERLIN GERIN	CER	PULSE	3
6	MERLIN GERIN	PM 710	RS485	3
7	MERLIN GERIN	PM 710	RS485	5
8	MERLIN GERIN	CER	PULSE	5
9	MERLIN GERIN	CER	PULSE	5
10	MERLIN GERIN	CER	PULSE	5
11	MERLIN GERIN	CER	PULSE	5
12	MERLIN GERIN	CER	PULSE	5
13	ABB	OD 4110	PULSE	5
14	REGULADORA	-	0	5
15	REGULADORA	-	0	5
16	MERLIN GERIN	CER	PULSE	5
17	REGULADORA	-	0	5
18	MERLIN GERIN	CER	PULSE	5
19	MERLIN GERIN	CER	PULSE	4
20	MERLIN GERIN	CER	PULSE	4
21	MERLIN GERIN	CER	PULSE	4
22	MERLIN GERIN	CER	PULSE	4
23	ABB	OD 4110	PULSE	4
24	ABB	OD 4110	PULSE	4
25	MERLIN GERIN	CER	PULSE	4
26	MERLIN GERIN	CER	PULSE	4
28	MERLIN GERIN	CER	PULSE	1
29	SIEMENS	D11R	0	1
30	MERLIN GERIN	CER	PULSE	1
31	MERLIN GERIN	CER	PULSE	1
32	MERLIN GERIN	CER	PULSE	1
33	ABB	OD 4110	PULSE	3
34	SIEMENS	D11R	0	3
35	SIEMENS	D11R	0	3

36	DUCATI	SMART PIU	RS485	3
37	DUCATI	SMART PIU	RS485	3
38	DUCATI	SMART PIU	RS485	3
39	DUCATI	SMART PIU	RS485	3
40	DUCATI	SMART PIU	RS485	3
41	DUCATI	SMART PIU	RS485	3
42	DUCATI	SMART PIU	RS485	3
43	DUCATI	SMART PIU	RS485	3
44	SIEMENS	D11R	0	2
45	SIEMENS	D11R	0	2
46	MERLIN GERIN	ME4 ZRT	PULSE	2
47	MERLIN GERIN	ME4 ZRT	PULSE	2
48	DUCATI	SMART PIU	RS485	3
49	MERLIN GERIN	ME4 ZRT	PULSE	2
50	CIRCUTOR	CVM MINI	RS485	2
51	MERLIN GERIN	PM 710	RS485	2
52	SIEMENS	D11R	0	2
53	SCHNEIDER ELECTRIC	A9 MEM 3210	PULSE	2
54	SIEMENS	D11R	0	2
55	SIEMENS	D11R	0	2
56	ABB	OD 4110	PULSE	2
57	MERLIN GERIN	CER	PULSE	2
58	MERLIN GERIN	CER	PULSE	2
59	ABB	OD 4110	PULSE	2
60	MERLIN GERIN	ME4 ZRT	PULSE	2
61	MERLIN GERIN	ME4 ZRT	PULSE	2
62	SCHNEIDER ELECTRIC	ME4 ZRT	PULSE	2
63	MERLIN GERIN	ME4 ZRT	PULSE	1
64	MERLIN GERIN	CER	PULSE	2
65	MERLIN GERIN	CER	PULSE	2
66	MERLIN GERIN	CER	PULSE	2

Anexo C

C.1 Protocolo Modbus RTU

Este protocolo impõe uma estrutura às mensagens trocadas entre dois ou mais aparelhos constituintes de uma rede. Um desses aparelhos terá de ser o *master*, e todos os outros aparelhos serão os *slaves*. O *master* é o aparelho responsável por realizar pedidos aos *slaves* e estes apenas respondem aos pedidos que lhes são feitos. Este protocolo pode usar vários meios de comunicação, mas o mais usual a nível industrial é o RS485. À semelhança deste protocolo, existe o *ModBus ASCII*, que em tudo é semelhante ao *RTU* tendo apenas algumas pequenas diferenças. O mais utilizado por aparelhos industriais é o *RTU*.

Como já explicado no paragrafo anterior e na secção 3.5.2, este protocolo pressupõe um master (mestre) e todos os outros aparelhos são slaves (escravos). Estes slaves respondem apenas a pedidos do master. Tanto estes pedidos como as respostas respeitam uma série de regras para que possam ser interpretadas pelo destinatário. Tanto um pedido como uma resposta são denominados de mensagens. Nas secções seguintes irá ser explicado como é construída uma mensagem neste protocolo.

C.1.1 Estrutura de uma mensagem

A estrutura de uma mensagem *ModBus RTU* é composta por vários campos como indicado na Tabela 6.7.

Tabela C.1 - Estrutura de mensagem Modbus RTU

Endereço	Função	Data	CRC
1 Byte	1 Byte	N x 1 Byte	2 Byte

Nota: 1 byte = 8 bits

C.1.2 Endereço

Cada aparelho na rede possui um número de 1 Byte que é a sua identificação na rede. Este número não pode ser repetido pois é o "Endereço" de cada aparelho. Para qualquer comunicação direcionada a um aparelho específico é preciso o seu endereço.

C.1.3 Função:

O *ModBus* disponibiliza uma série de funções que o master pode pedir para o *slave* realizar, entre as quais, ler um registo, forçar o valor de uma saída digital, etc. Neste campo é onde é colocado o código da função que é codificada em 8 bits.

C.1.4 Funções existentes na norma ModBus

01 READ COIL STATUS (Lê estado de memória booleanas.
Nota: 1 memória booleana = 1 bit

02 READ INPUT STATUS (Lê o valor de entradas)

03 READ HOLDING REGISTERS (Lê o valor de memórias retentivas)

04 READ INPUT REGISTERS (Lê o valor de registos de entrada)
Nota: 1 registo = 16 bits, 2 bytes)

05 WRITE SINGLE COIL (Edita o estado de uma memória booleana)

06 WRITE SINGLE REGISTER (Edita o valor de 1 registo)

15 WRITE MULTIPLE COILS (Edita o estado de várias memórias booleanas)

16 WRITE MULTIPLE REGISTERS (Edita o valor de vários registos)

C.1.5 Data:

Aqui é onde é escrita a informação necessária para realizar as mais variadas funções como por exemplo a posição de memória a ler e o número de posições consecutivas. Este campo depende da função e se se trata de um pedido ou resposta.

C.1.5 CRC:

Conhecido como CRC o "*cyclic redundancy check*" é um número de 2 Bytes que resulta de um algoritmo matemático que utiliza os restantes Bytes da mensagem. Desta forma o CRC é um código que pode ser utilizado para verificar se a informação enviada é a mesma que a recebida.

Nota: Normalmente, as representações de mensagem *ModBus RTU* são feitas com caracteres hexadecimais, pois todos os campos têm tamanhos múltiplos da unidade *Byte* e **um caracter hexadecimal precisa de 4 bits para ser codificado**. Assim, ao longo do documento **todos os campos de mensagens *ModBus RTU* vão estar representados por um número múltiplo de 2 caracteres hexadecimais (2caracteres = 8bits = 1Byte)**.

Para percebermos melhor o significado destes campos, vamos ver um exemplo de uma transação específica:

C.1.6 Pedido:

Na Tabela 6.8 está representado um pedido ModBus RTU.

Tabela 6.1: Pedido enviado

Endereço do *slave* (1 Byte)	Função (1 Byte)	Posição de começo (2 Bytes)	Quantidade de endereços (2 Bytes)	CRC (2 Bytes)
04	01	00 0A	00 0D	DD 98

Endereço do *slave:* Endereço do aparelho a que o pedido se destina. Neste caso é o aparelho número 4.

Função: Código da função, neste caso trata-se de uma função para ler o estado de memórias com apenas um bit, conhecidas como *Coils*. A função com o código número 1 na norma *ModBus* tem a o nome de *"Read Coils"*

Posição de começo: Endereço da posição de memória onde é pretendido fazer a leitura. Neste caso quer-se ler a partir da posição 10 inclusive (A em hexadecimal).

Quantidade de endereços: Número de posições de memória que é desejado ler, que são 13 (D em hexadecimal)

CRC: Verificação CRC.

C 1.7 Resposta

A resposta ao pedido efetuado acima está representada na Tabela 6.2.

Tabela 6.2: Resposta ao pedido

Endereço do *slave* (1 Byte)	Função (1 Byte)	Contagem de Bytes (1 Bytes)	Informação dos endereços (2 Bytes)	CRC (2 Bytes)
04	**01**	**02**	**0A 11**	**B3 50**

Endereço do *slave:* Endereço do aparelho a que pertence a mensagem de resposta.

Função: Código da função a que o *slave* está a responder.

Contagem de Bytes: Número de Bytes que o campo **Informação dos Endereços** contém.

Informação dos Endereços: Informação pedida pelo master. $0A\ 11_{16} = 0101000010001_2$. Cada bit da resposta representa o estado de cada *coil*, sendo o bit da esquerda o referente à posição de memória mais significativa.

CRC: Verificação CRC.

Anexo D

D.1 Modbus RTU no Fatek FBs-20-MC (com expansão FBs-CM25E)

D.1.1 Objetivo

- Fazer comunicação entre o *Fatek FBs20MC* e o analisador de energia *Janitza UMG 604*.
- Comunicar através de uma aplicação *Visual Basic* e o autómato *Fatek FBs20MC.*

D.1.2 Resumo

Este trabalho tem por objetivo realizar um teste de modo a simular a comunicação entre um computador e uma rede de autómatos/PLC's que por sua vez comunicam com analisadores/contadores de energia como ilustrado na Figura 6.1. Estes contadores podem possuir comunicação *RS485 Modbus RTU*, ou apenas uma saída de impulso, que será utilizada como entrada digital no autómato. Assim, para este teste a rede de autómatos resume-se apenas a um autómato *Fatek FBs-20MC* ilustrado na Figura 6.2 e o analisador *Janitza UMG 604* ilustrado na Figura 6.4. Quanto à saída de impulso será simulada com um botão de pressão associado à entrada X0 do autómato.

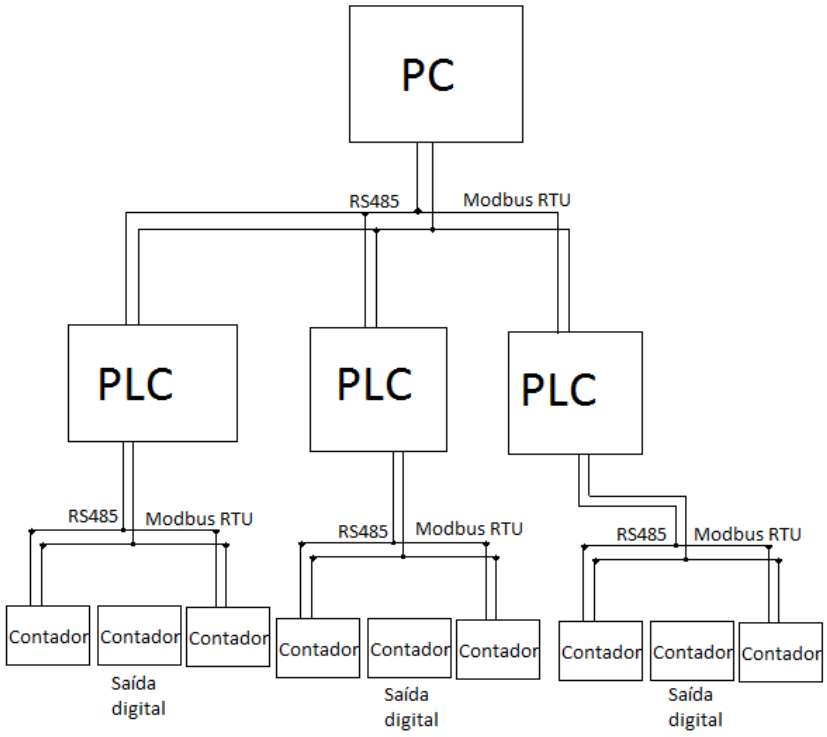


Figura 6.1: Imagem ilustrativa da arquitetura a simular

Neste trabalho vai ser realizada uma comunicação *RS485* utilizando o protocolo *Modbus RTU* entre o *Fatek* e o *Janitza*. A interface *RS485* encontra-se numa expansão (*FBs-CM25E*) ilustrada na Figura 1.3, que este autómato possui. Nesta ligação o autómato terá de atuar como o *master* (mestre) da ligação enquanto o analisador de energia atuará como *slave* (escravo).

Pela interface *RS232* também presente na expansão, o autómato irá comunicar com um computador através de uma aplicação *Visual Basic*. Nesta ligação o autómato será o *slave* e o computador será o master.

A interface denominada *Port0* será apenas para a programação do autómato.

O protocolo escolhido para ambas as ligações foi o protocolo *Modbus RTU*, pois é amplamente utilizado por aparelhos industriais que efetuam comunicação. Além disso evita-se utilizar um protocolo proprietário da *Fatek*, o que limitaria uma possível comunicação com outros autómatos de outras marcas.

Desta forma o computador terá acesso indiretamente à informação que o analisador de energia contém.

Por fim, para simular uma saída de impulso será utilizado um botão de pressão.

D.1.3 Material utilizado

- Fatek FBs-20MC

Figura 6.2: Imagem ilustrativa do autómato utilizado [35]

- FBs-CM25E

Figura 6.3: Imagem Ilustrativa da expansão utilizado com o autómato [36]

- Janitza UMG 604

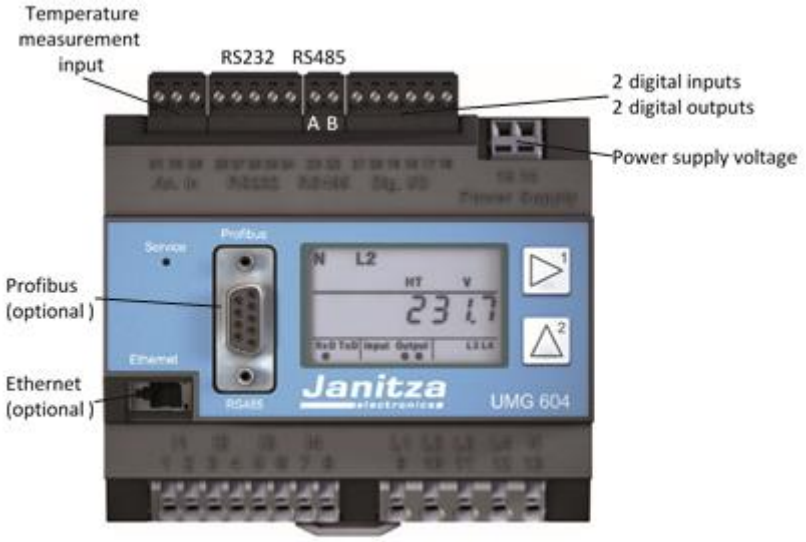


Figura 6.4: Aparelho Janitza UMG604 (adaptado de [37]*)*

D.1.4 Implementação do protocolo Modbus RTU master na interface PORT4 (RS485)

Tão ou mais importante do que implementar um protocolo é a configuração da ligação numa porta série. Assim antes de definir o protocolo, irão ser definidos os parâmetros da ligação. Os parâmetros escolhidos foram os seguintes: **9600 bps, 8 data bits, No Parity, 1 stop bit.**

Para definir estes parâmetros é necessária a consulta do **capítulo 12 do manual do autómato** [10]**, secção 12.4.3 (página 14 e 15)**. Da consulta obtêm-se a seguinte informação:

R4044: -1º byte=56H

-2º byte=0 (Even Parity [é indiferente]) + 1 (8 data bits) + 0 (None Parity) + 1 (para 1 stop bit) + 0001 (9600 bps)

Então é necessário colocar o valor **5641H no registo R4044**, para aplicar os parâmetros requeridos.

Aplicados os parâmetros da comunicação, vai-se agora implementar o protocolo. Basta para isso utilizar a função **150.M-Bus**, ilustrada na Figura 1.5 e explicada a partir da **página 40 do capítulo 13 do manual** [38], para assim enviar mensagens *Modbus RTU*, e receber a hipotética resposta.

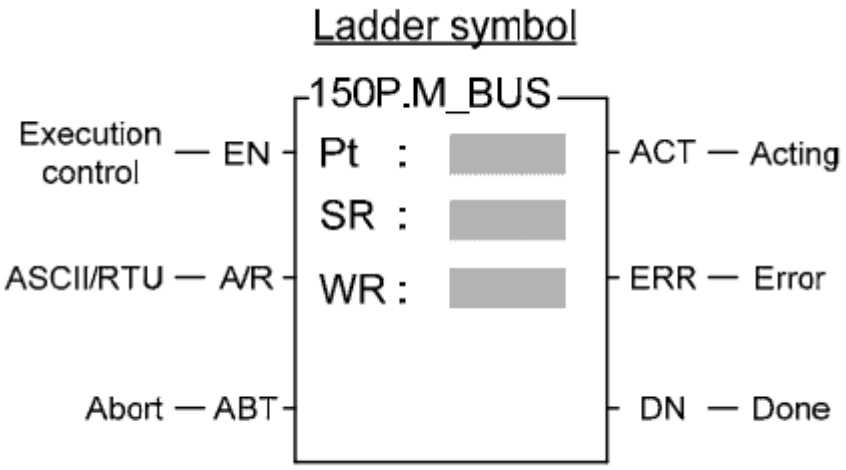


Figura 6.5: Imagem Ilustrativa da função 150.M-Bus (adaptado de [38]*)*

Ao utilizar esta função é necessário definir 3 valores. *Pt* é a porta que irá transmitir a informação. Neste caso é a 4. *SR* é o registo inicial onde é colocada a informação que vai conter a mensagem *Modbus*. E *WR* é registo inicial onde irá ser escrita a informação de como decorreu a ação (a ser preenchido pelo autómato).

Na **página 45 do capítulo 13** [9], está explicado o significado de cada registo *SR*, e que é detalhado a segui:

Definindo o *SR* como sendo *R0*, por exemplo:

- **SR+0=R0: A550H** (valor predefinido a colocar no 1º registo)
- **SR+1=R1: 1º BYTE=07H, 2BYTE=nº** de mensagens *Modbus* que a enviar. Isto é: realizando um pedido com a função 04 e fazendo o *set* de uma saída com a função 05, ou então realizando o pedido do valor de 2 posições de memória não consecutivas por exemplo, este campo tem de ter o valor **2**.

A partir deste registo, vamos escrever em **7xN registos sendo N o 2byte do registo R1**. Ou seja, cada mensagem e respetiva resposta, precisa e 7 registos para ser configurada.

- **SR+2=R2**: nº do *slave*

- **SR+3=R3**: código do comando a realizar. Atenção! Este código não é o mesmo que é definido pela norma *Modbus RTU*, mas sim pelo *software da Fatek*. Assim sendo, temos os seguintes valores: 1-ler informação do *slave*, 2-escrever múltipla informação no *slave*, 3-escrever num único registo
- **SR+4=R4**: nº de posições a ler no *slave*
- **SR+5=R5**: tipo de dados do *master (PLC)* onde irá ser escrita a informação que venha na resposta. O nº associado a cada tipo de dados encontra-se na **página 46**.
- **SR+6=R6**: Nº do registo onde irá ser escrita a informação. Se queremos que escreva a informação no registo R100, então **R6=100**
- **SR+7=R7**: tipo de dados do *slave*. O nº correspondente encontra-se também na **página 46 do capítulo 13**.
- **SR+8=R8**: Neste registo colocamos o endereço do valor que queremos ler no *slave*.

Quanto aos registos do tipo WR, é onde vai ser escrita a informação de como correu a comunicação. Ocupam 8 registos e a informação detalhada encontra-se na **página 46, capítulo 13 do manual.**

D.1.5 Implementação do protocolo Modbus RTU slave na interface PORT3 (RS232)

Por defeito as portas do autómato estão com o protocolo *Fatek*. Para alterar esta definição, podemos utilizar o registo R4047, como indicado na **secção 12.4.2, do capitulo 12 do manual do autómato**. Para que a porta 3 utilize o protocolo *Modbus RTU slave*, um dos valores possíveis é R4047=5620H.

Assim sendo desta simples forma, a porta 3 passa agora a interpretar o protocolo *Modbus RTU*, mas como sendo um *slave* da rede.

No entanto, existe um problema. À exceção das funções 5 e 6 do protocolo *Modbus RTU*, o autómato não está preparado para interpretar as restantes funções. Para isso é preciso fazer um "mapeamento da memória" como descrito e explicado na **página 50 do capítulo 13**. Consultando a tabela presente nessa página e fazendo o mapeamento por exemplo para a função 04, é evidente que é necessário editar os registos R3968, R3975, R3976, R3977.

- R3968=A55AH, significa que o utilizador vai utilizar um novo mapeamento dos endereços para a comunicação com o protocolo *Modbus RTU slave*.
- R3975= valor de início do endereço *Modbus RTU* utlizado pelo master.
- R3976= valor de início do endereço referente à gama de registos do tipo R que vão ser lidos.
- R397= intervalo de registos que vamos ler.

Isto é, se R3975=50, R3976=70, R3977=100, então, é possível ler um intervalo de 100 registos do tipo R, que vão desde R70 a R170. O *master* para ler estes registos, pode enviar posições que variam do 50 até 150, ou seja, por exemplo, o valor 60 do master refere-se ao R80.

D.1.6 Valores a ler no aparelho Janitza

Os valores a ler no aparelho *Janitza* são os seguintes:

- IP do aparelho (Endereço *Modbus* no aparelho: 10190)
- Energia Fornecida Real (Endereço *Modbus* no aparelho: 9850)
- Valor do número de impulso do botão de pressão

A função 150.M-bus do *Winproladder* retira 1 unidade aos endereços definidos, pois existem 2 conceitos na norma *Modbus*. A definição de nome, que normalmente é o valor que vem no manual do aparelho referente a uma certa posição de memória, e a definição de endereço que não é nada mais do que o valor do nome subtraído de uma unidade. Neste caso o manual do aparelho *Janitza* fornece os endereços e não os nomes. Mas como a função 150.M-bus espera os nomes, temos de somar uma unidade os valores retirados do manual do aparelho.

Assim sendo os valores que terão de ser introduzidos no programa são:

- IP do aparelho: 10191
- Energia Fornecida Real: 9851

D.1.7 Implementação do programa Ladder

Na Figura 6.6, está a página principal do programa *Ladder*, que é responsável por correr a função 150.M-bus a cada segundo e verificar a cada ciclo se alguém premiu o botão.

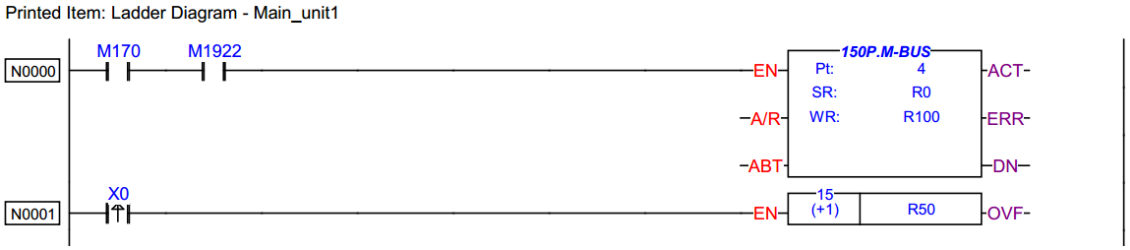


Figura 6.6: Cópia de parte do programa Ladder

De notar que no *SR* (*start register*), foi colocado o valor *R0*. Como já foi referido é nestes registo que é colocada a informação da estrutura da(s) mensagens *Modbus RTU*, a ser(em) enviada(s).

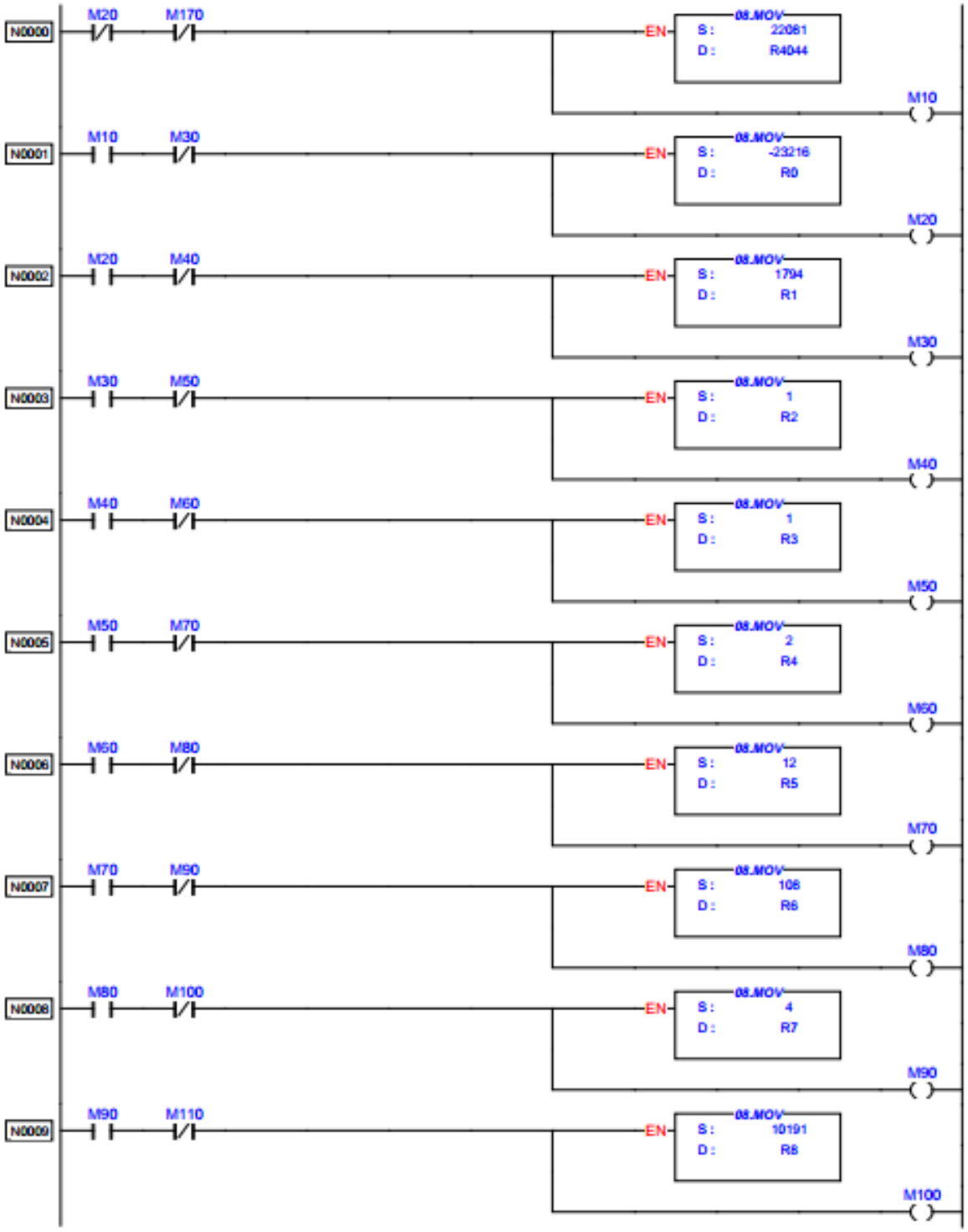
Como pode ser observado na Figura 6.7, os valores dos registos são os seguintes:

- R0: -23216 = A550H, valor predefinido que indica a utilização da função 150.M-Bus
- R1: 1794 = 0702H, 1º byte é predefinido, o 2º indica o nº de pedidos que vamos realizar ao *slave*
- R2: 1, endereço do *slave*
- R3: 1, função a realizar segundo o *Winproladder* (=1, ler informação do *slave*)
- R4: 2, comprimento (em *words*) da resposta a receber
- R5: 12, tipo de dados do registo do PLC para guardar a resposta segundo o *Winproladder* (=12 R(*data register*))
- R6: 108, endereço do registo inicial onde irá ser guardada a resposta (R108)
- R7: 4, tipo de dados do *slave* segundo o *Winproladder* (=4, *Holding Register*)
- R8: 10191, endereço a ler do *slave*

Daqui em diante os registos repetem-se à imagem do registo R2 inclusive em diante, para configurar o segundo pedido

- R9: 1, endereço do *slave*
- R10: 1, função a realizar
- R11: 1, comprimento da resposta
- R12: 12, tipo de dados, do *start register* para guardar a resposta, do PLC
- R13: 110, endereço do *start register* para guarda a resposta
- R14: 4 tipo de dados do *slave*
- R15: Endereço da posição de memória ler do *slave*

Printed Item: Ladder Diagram - config_Pt4_contador_Modbus



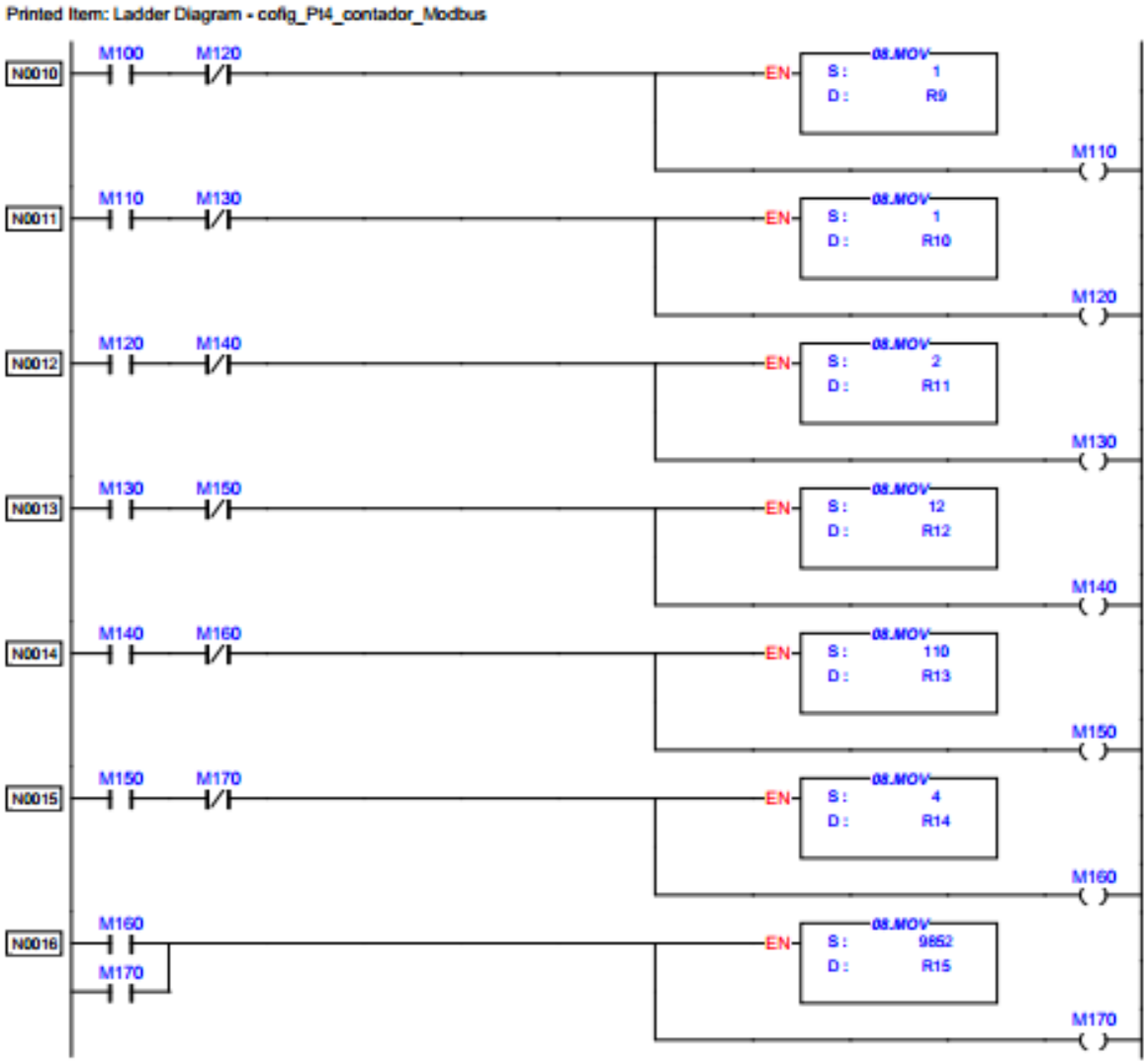


Figura 6.7: Cópia de parte do programa Ladder

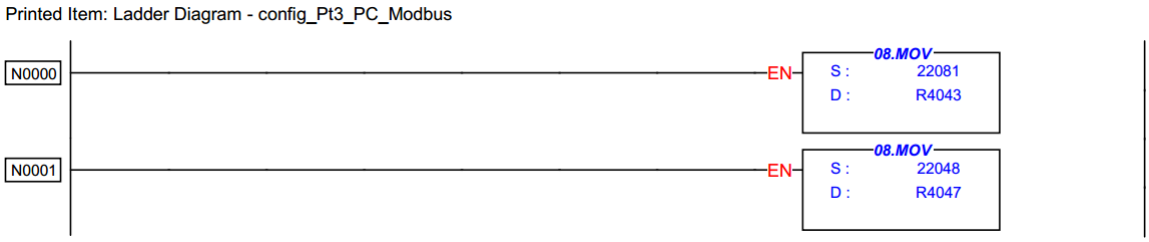


Figura 6.8: Cópia de parte do programa Ladder

D.1.8 Programação do PLC e conexão da rede

Com os conhecimentos da parametrização e configuração das diferentes ligações do autómato, é possível agora montar todo o sistema e programar o PLC.

D.1.9 Programa PLC

O programa PLC encontra-se em anexo.

D.1.10 Ligações

Então, para a simulação da rede, será utilizado apenas um PLC e 1 analisador. A saída digital poderá ser um botão de pressão.

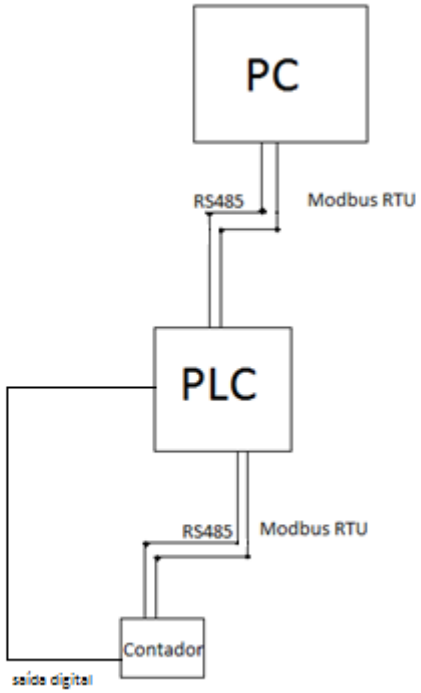


Figura 6.9: Imagem ilustrativa da simulação efetuada

D.1.11 Leitura das posições de memória do PLC

Com um programa desenvolvido em Visual Basic (Figura 6.10), vão ser lidas as posições de memória em que foram guardados os valores lidos do analisador de energia.

Devido ao mapeamento que foi feito, o autómato irá responder da seguinte maneira aos endereços *Modbus* seguintes:

- 110 = (00 6E) H - Endereço inicial onde é guardado o valor da Energia Fornecida Real
- 108 = (00 6C) H – Endereço inicial onde é guardado o valor do IP
- 50 = (00 32) H – Endereço onde é guardado o valor dos impulsos

Na Figura 6.10, é mostrada a resposta ao pedido *Modbus* do valor de impulsos. É possível ver que a resposta dada é: (00 4E) H = 78 impulsos.

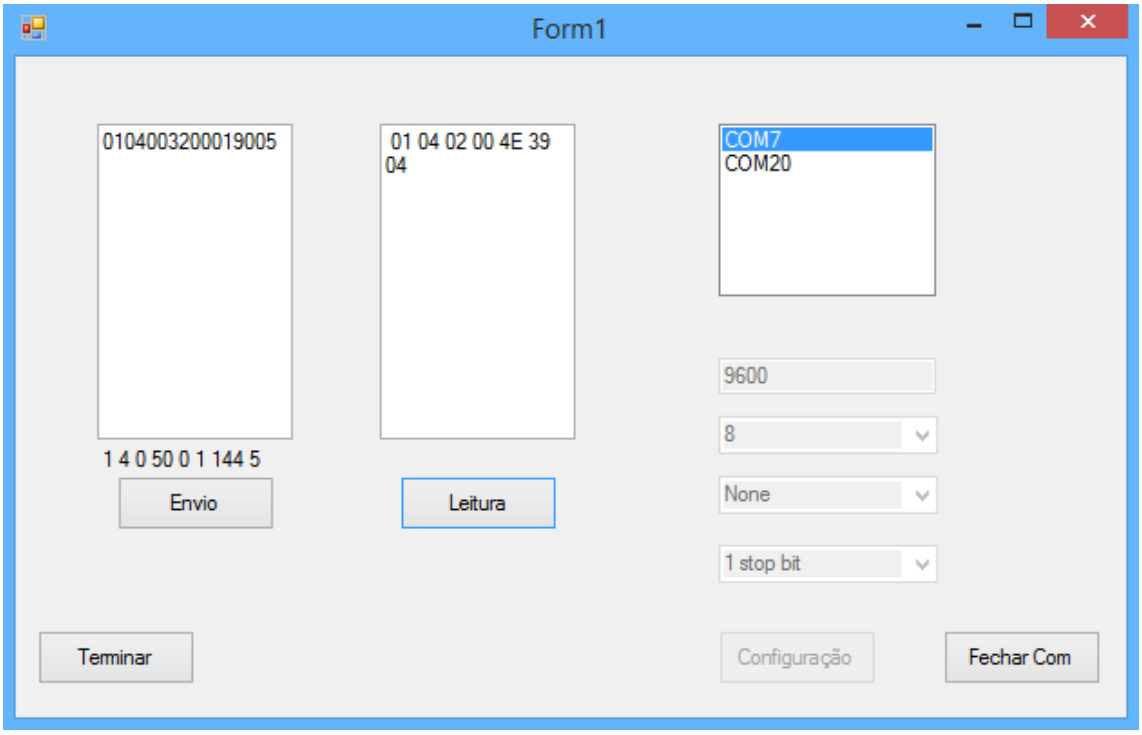


Figura 6.10: Ilustração do programa VB

Na Figura 6.11, é mostrada a resposta ao pedido *Modbus* do valor da energia real fornecida. É possível ver que a resposta dada é: (48 A1 B1 DF) H = 331150,97 Wh

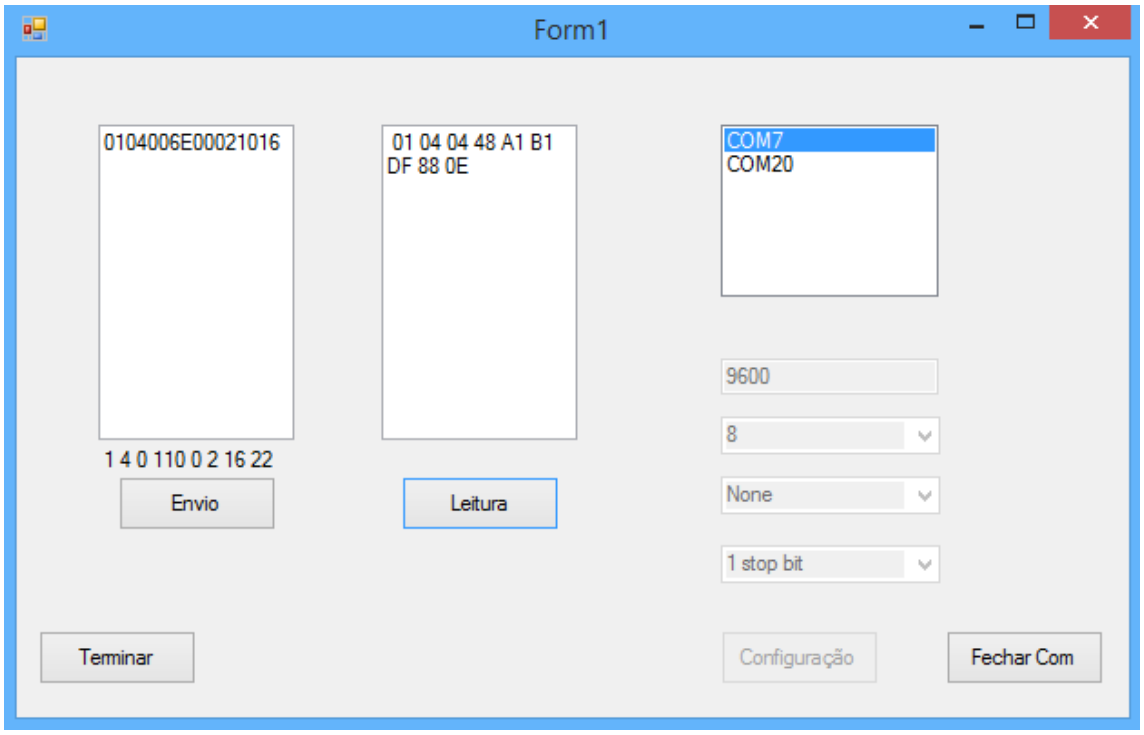


Figura 6.11: Ilustração do programa VB

Na Figura 6.12, é mostrada a resposta ao pedido *Modbus* do valor do IP. É possível ver que a resposta dada é: (C1) H = 193, (89) H = 137, (AC) H = 172, (17) H = 23.

Assim a resposta é: 193.137.172.23

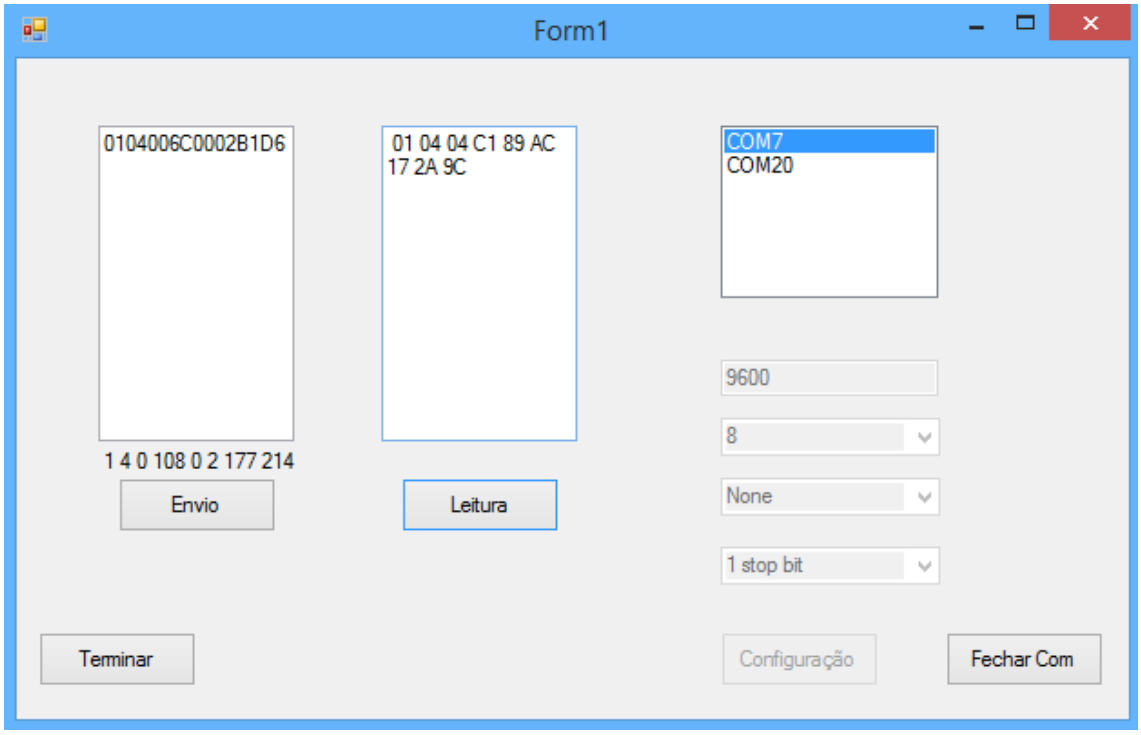


Figura 6.12: Ilustração do programa VB

Anexo E

E.1 Comunicação de um analisador de energia Janitza com o software Movicon

E.1.1 Objetivo

Fazer a comunicação entre o *software SCADA Movicon* e o Analisador de energia *Janitza UMG 604*. Através desta comunicação, o *Movicon* será capaz de receber valores medidos pelo aparelho *Janitza*, como a potência reativa de uma fase ou a diferença de potencial. A Figura 1.1 ilustra este teste.

E.1.2 Resumo

Para realizar este teste, temos um computador onde temos instalado o *software Movicon*, o aparelho *Janitza* e um adaptador *USB-RS485*.

A ligação entre o computador e o aparelho e feita através de RS485, utilizando o protocolo *Modbus RTU.* Este tipo de ligação foi o escolhido pois está disponível tanto no analisador como no *software*. Além disso trata-se de uma ligação muito utilizada em meio industrial pois é bastante imune a interferências eletromagnéticas quando comparado com outras comunicações série como o RS232 por exemplo.

Neste exemplo em particular vamos comunicar com o aparelho para que este nos responda o seu IP.

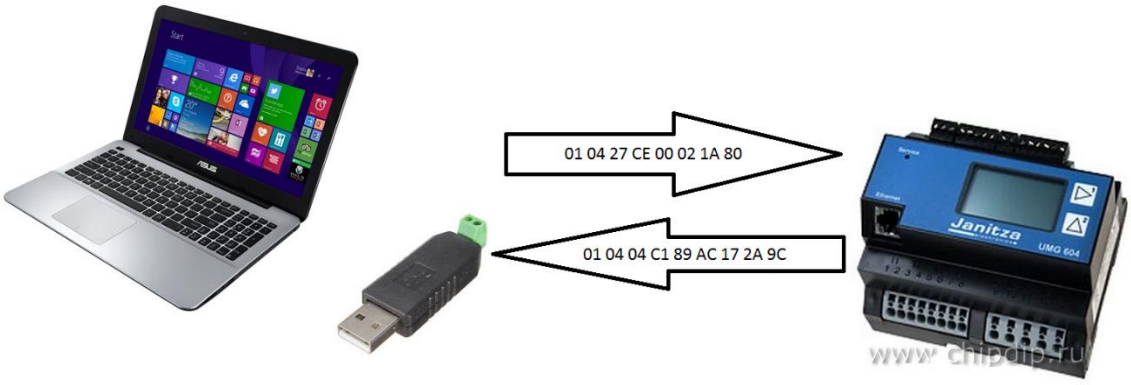


Figura 6.13 Imagem ilustrativa do teste realizado (adaptado de [39][40][41]*)*

E.1.3 Material

- Analisador de energia Janitza UMG604
- Conversor USB-RS485 (chip PL-2303 HX)
- Software Movicon 11.4

E.1.4 Analisador de energia, o que é e para que serve?

Um analisador de energia serve para medir vários parâmetros de um sistema elétrico. Com um aparelho deste género podemos por exemplo determinar a potência que um sistema elétrico consome, a sua intensidade de corrente, potência recativa etc. Estes aparelhos são úteis para quantificar perdas de energia num sistema para posteriormente redimensionar e tentar por exemplo, corrigir fatores de modo a aumentar a eficiência energética.

E.1.5 Janitza UMG604

O Janitza UMG604 é concebido para medição e cálculo de grandezas elétricas. Os resultados das medições podem ser exibidos no ecrã, armazenados e lidos posteriormente via comunicação série. Este aparelho apresenta algumas das características seguintes:

- 4 *inputs* para medir a diferença de potencial
- 4 *inputs* para medir a diferença de corrente
- Medição contínua de corrente e diferença de potencial dos *inputs*
- Análise de Fourier de 1ª até 40ª ordem, das components harmonicas para U, I, P (consumo/fornecido) e Q (inductive/capacitive)
- 2 digital inputs,
- 2 digital outputs,
- LED display,
- 2 botões
- RS485 (modbus RTU, modbus master),
- RS232,
- Ethernet
- Programação de aplicações próprias em Jasic
- Temperatura de funcionamento -10°C.. +55°C

Algumas destas características estão indicadas na Figura 6.14 referente ao analisador de energia em causa.

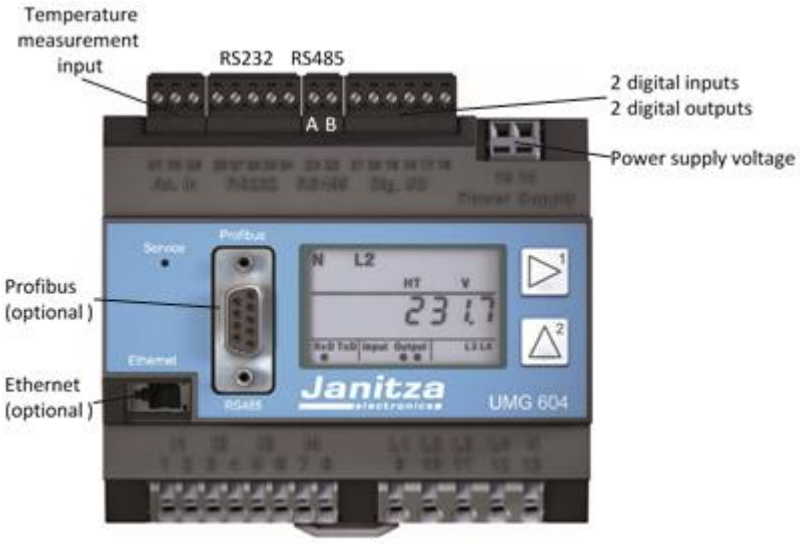


Figura 6.14 Aparelho Janitza UMG604 [37]

E.1.6 Ligações do Janitza

Para se fazer medições de um sistema elétrico com o analisador Janitza UMG604, tem de haver consumo de energia elétrica por parte de um aparelho. Na figura 1.3, é mostrado o exemplo de um sistema constituído por um motor elétrico trifásico.

As ligações à rede para o analisador Janitza fazem-se de acordo com a Figura 6.15:

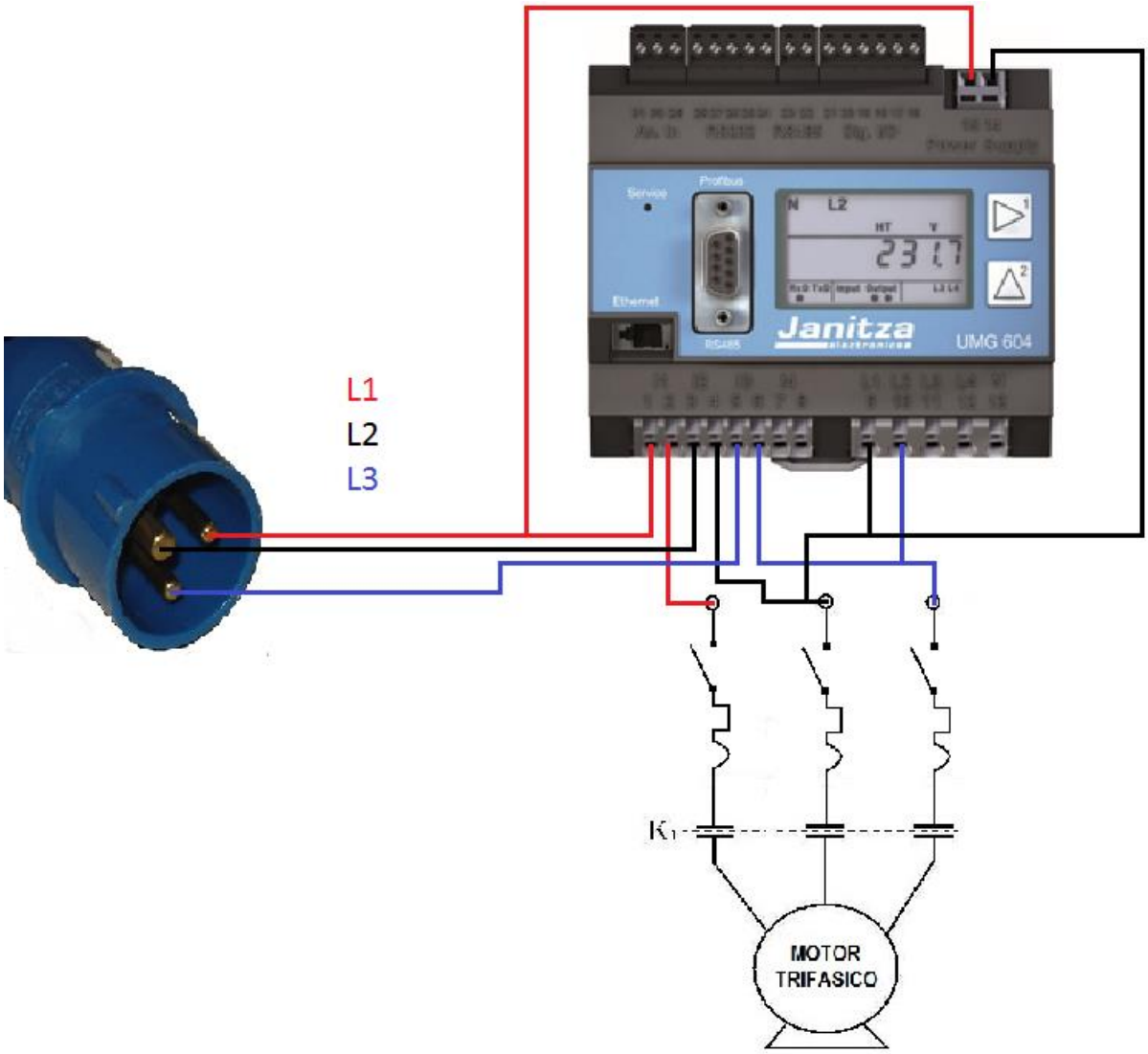


Figura 6.15 Ilustração de uma possível ligação do analisador Janitza a um sistema elétrico ([37], [42], [43]*)*

Quanto às ligações da comunicação série, temos a figura 1.4 que exemplifica para o caso do RS485:

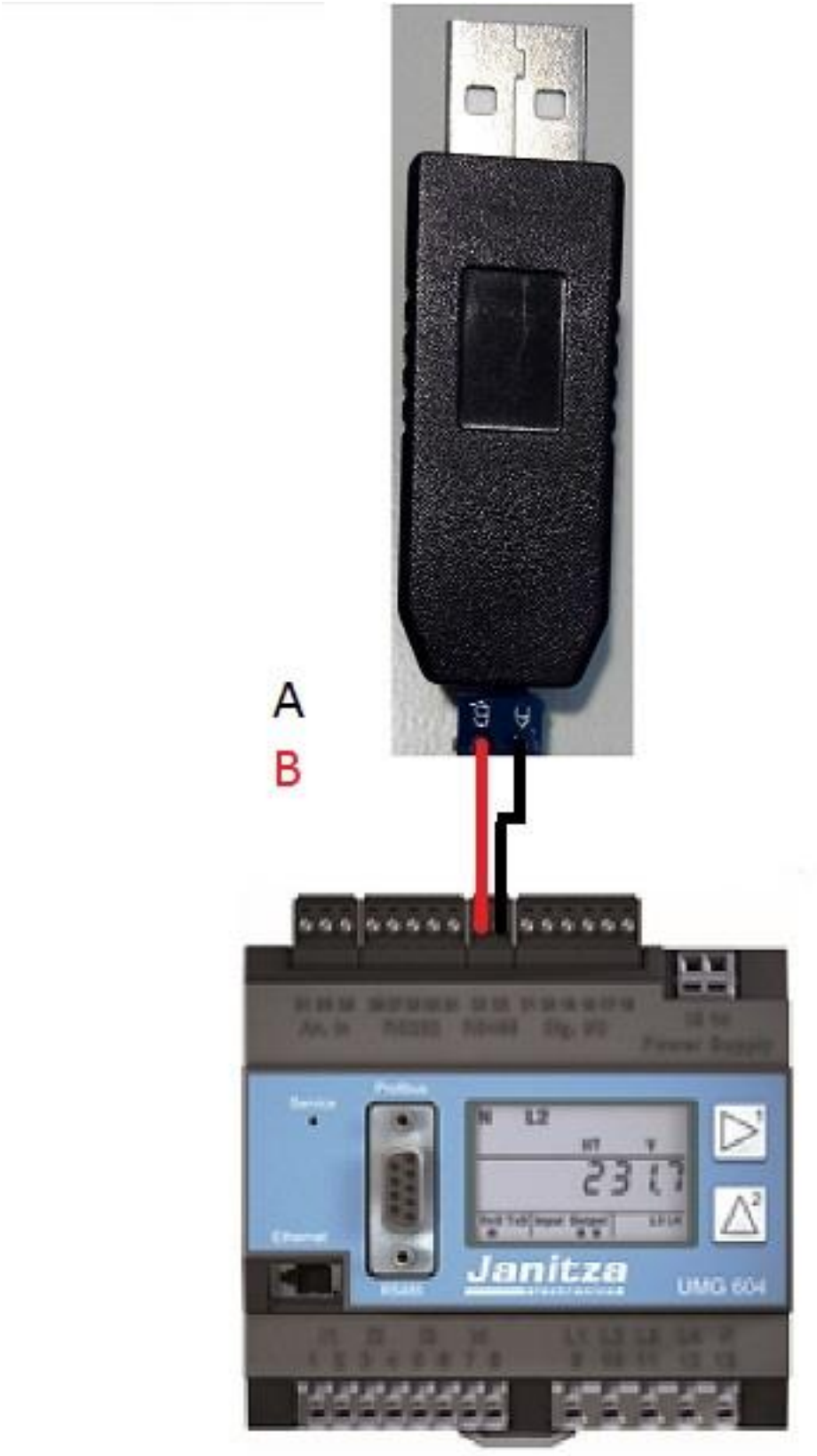


Figura 6.16 Ilustração da ligação do analisador ao conversor RS485 (adaptado de [37]*)*

Como se pode ver na Figura 6.16, o A e B referentes à interface RS485, estão ligados de forma cruzada, ao A e B do conversor USB.

E.1.7 Protocolo Modbus RTU

O protocolo *Modbus*, é uma estrutura de mensagem utilizada em comunicação série e pressupõe um *master* e um ou mais *slaves*. Esta estrutura de mensagem prevê vários campos que vamos no texto a seguir. A mensagem *Modbus* é constituída por caracteres hexadecimais.

E.1.8 O pedido

O pedido é uma mensagem enviada pelo *master* para um ou mais *slaves*. Esta mensagem tem entre outras informações, o endereço do(s) *slave(s),* a função que informa o *slave* o tipo de ação vai ser realizada, posição de memória inicial, número de posições de memória a aplicar a função, etc.

E.1.9 A resposta

Caso o pedido por parte do *master*, exija uma resposta por parte do *slave*, este tem de responder. A resposta contém a função a que está a responder, a informação associada, etc.

E.1.10 Modo RTU

No modo *RTU*, cada caracter da mensagem *Modbus*, é codificado em 4*bits*, assim, cada *byte*, representa 2 caracteres hexadecimais.

E.1.11 Estrutura das mensagens - Pedido

Tabela 6.3: Estrutura de uma mensagem Modbus RTU master

Endereço	Função	Endereço Inicial dos Registos		Quantidade de Registos		CRC	
1 *byte*	1 *byte*	1 *byte*	1 *byte*	1 *byte*	1 *byte*	1 *byte*	1 *byte*

E.1.12 Estrutura das mensagens - Resposta

Tabela 6.4: Estrutura de uma mensagem Modbus RTU slave

Endereço	Função	Contagem dos Bytes	Informação		CRC	
1 *byte*	1 *byte*	1 *byte*	1 *byte*	1 *byte*	1 *byte*	1 *byte*

E.1.13 Teste com o programa Realterm

Na Figura 6.17, temos a imagem de um programa chamado *Realterm*, que serve para comunicar com portas série. Nesta imagem temos sublinhada a mensagem enviada e a seguir a mensagem que o analisador Janitza respondeu.

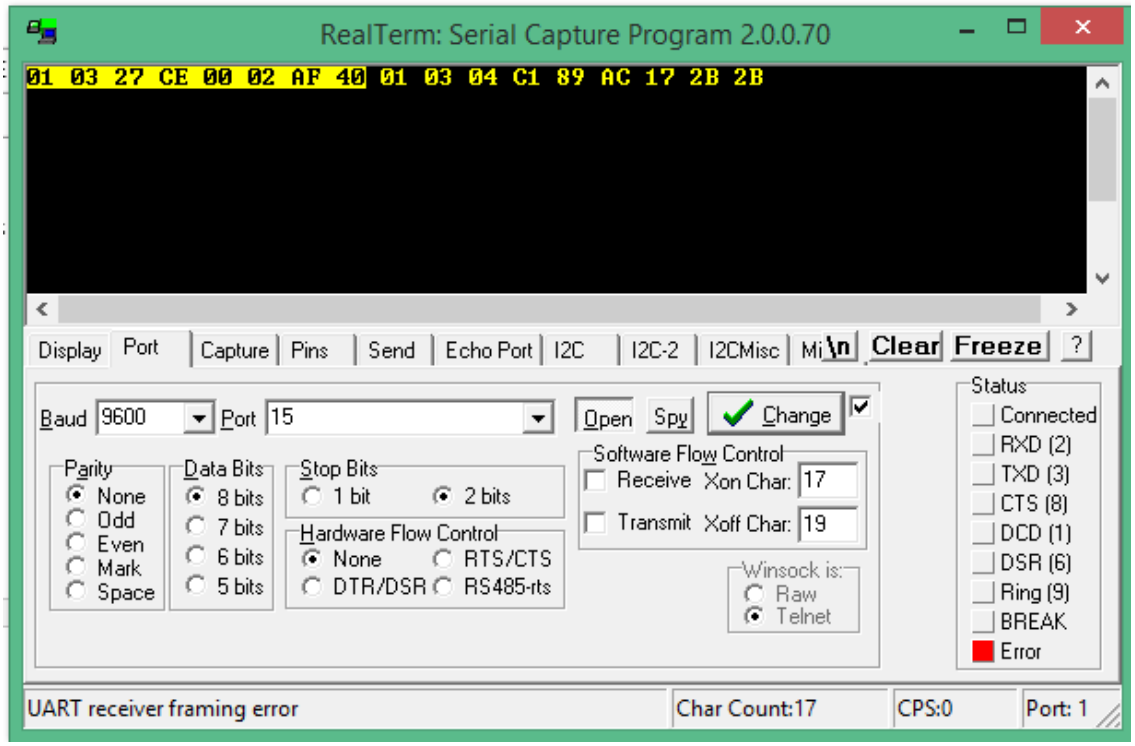


Figura 6.17: Programa Realterm*, envio e receção da resposta*

Para entender melhor o teste representado acima vamos ver como é constituída uma palavra *Modbus RTU* para este caso específico.

Tabela 6.5: Estrutura da mensagem Modbus RTU master *enviada no exemplo*

Endereço[1]	Função[2]	Endereço Inicial dos Registos[3]		Quantidade de Registos[4]		CRC[5]
01	03	27	CE	00	02	AF 80

As propriedades a configurar são as seguintes:

- 1 – Foi definido quando foi feita a configuração da ligação (Figura 6.17)
- 2 - *Data Area* – para este exemplo trata-se de um *Input Register*
- 3 - *Start Adress* – é o endereço de registo inicial, neste caso, segundo o manual do Janitza é o *10190* em decimal
- 4 - *# Elements* – Quando colocado a 0, é feito o cálculo automático do número de registos a ler
- 5 – É calculado automaticamente pelo *Movicon*
- *Type = Input*

Nota: Para obtermos a palavra devidamente ordenada, temos de colocar *Swap Bytes= True* e *Swap Words= True*.

Enviada esta palavra obtemos esta resposta:

Tabela 6.6: Estrutura da mensagem Modbus RTU slave *enviada no exemplo*

Endereço[1]	Função[2]	Número de bytes [6]	Resposta[7]	CRC[5]
01	03	04	C1 89 AC 17	2B 2B

- 6 - Número de bytes que a resposta contém
- 7 – resposta: $C1.89.AC.17_{16} = 193.137.172.23_{10}$

A resposta está correta. Vamos agora em seguida fazer este mesmo teste mas tendo como *master*, um programa *Movicon.*

E.1.14 Procedimento

Depois de instalar o programa *Movicon*:

Abrir o programa *Movicon*, iniciar um novo projeto utilizando o menu *File* e selecionando a opção *New* como mostra a Figura 6.18.

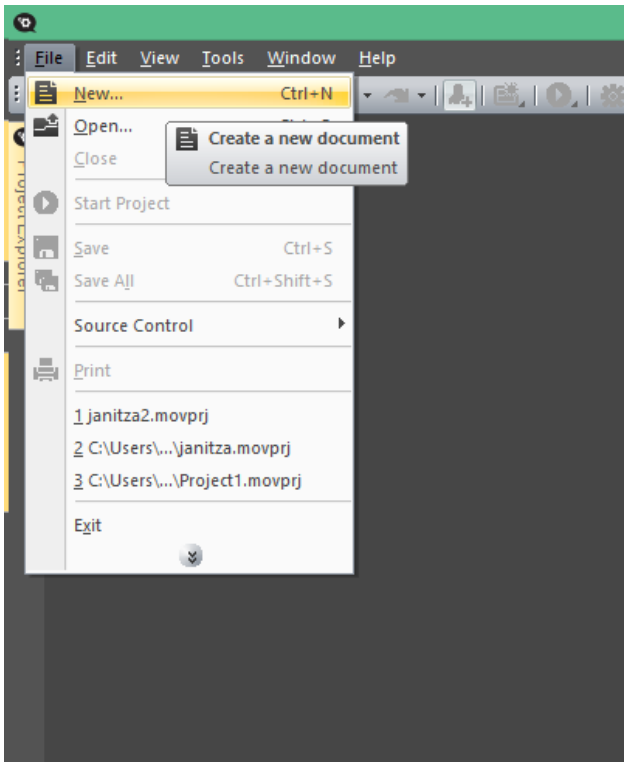


Figura 6.18: Criação de um novo projeto no software Movicon

Selecionar o tipo de aplicação *Win32/64 platform* como na Figura 6.19.

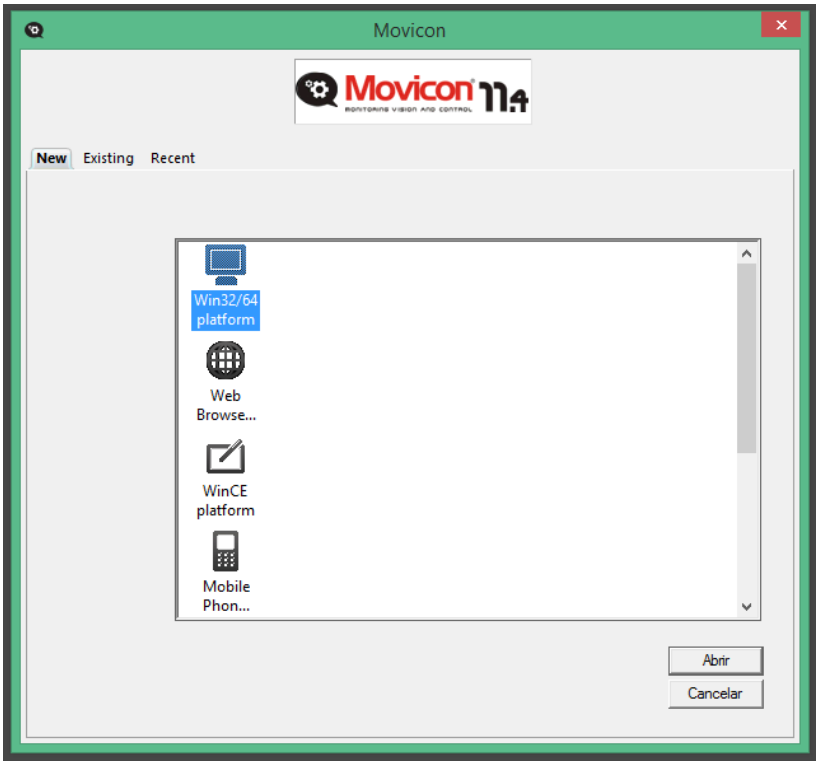


Figura 6.19: Escolha do tipo de aplicação a desenvolver no software Movicon

Dar nome ao projeto e a sua localização como mostra a Figura 6.20.

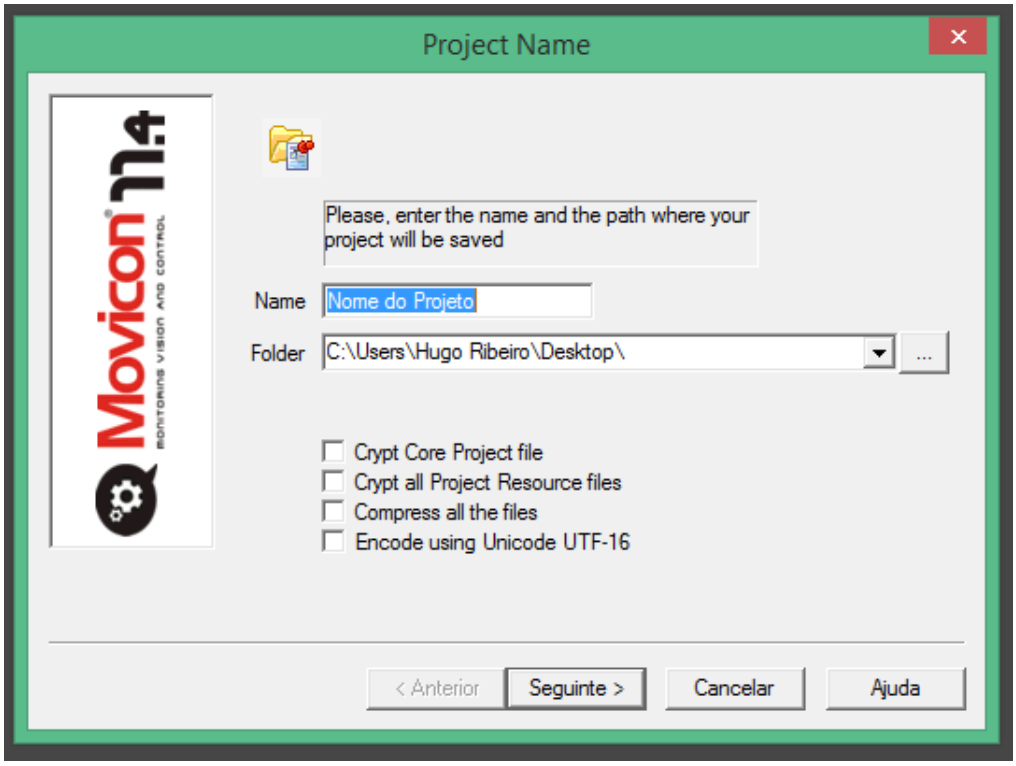


Figura 6.20: Escolha do nome e localização na criação de um novo projeto no software Movicon

Selecionar o *drive* para a comunicação como indicado na Figura 6.21.

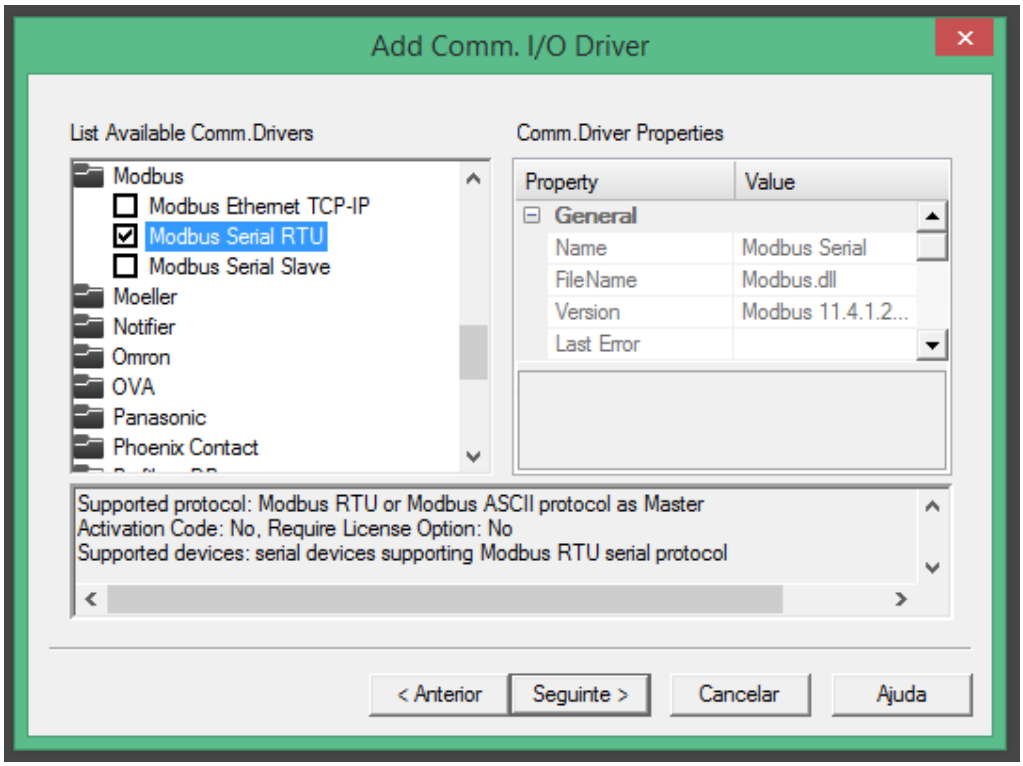


Figura 6.21: Escolha de drives *de comunicação* Modbus RTU *de um novo projeto no* software Movicon

Adicionar uma ligação à imagem da Figura 6.22.

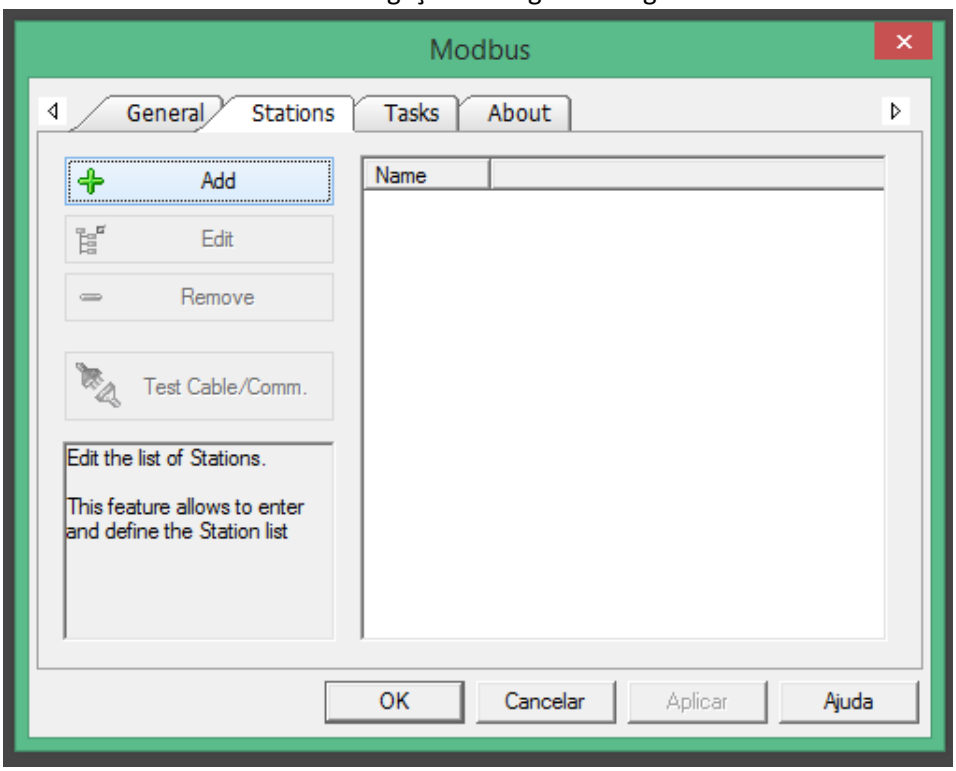


Figura 6.22: Adição de uma nova ligação no software Movicon

Configurar a ligação como representado na Figura 6.23.

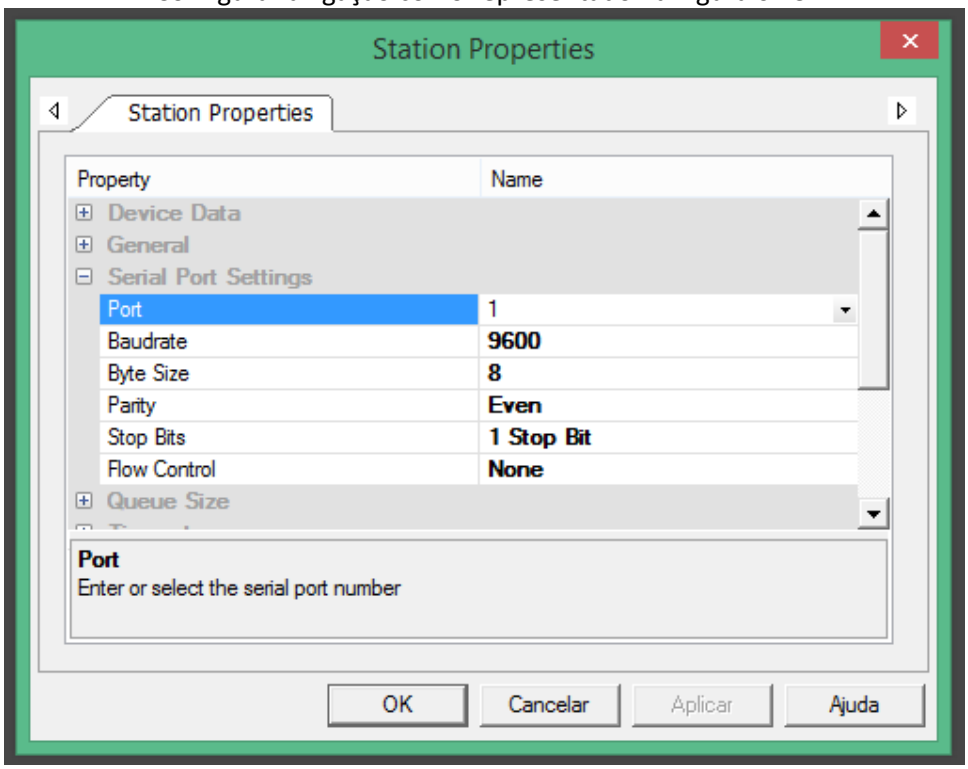


Figura 6.23: Configuração de uma ligação no software Movicon

Nota: O analisador de energia terá de ter obviamente os mesmos parâmetros para a ligação RS485.

Com a ligação configurada é preciso agora configurar um pedido. Neste exemplo será o pedido do IP. Um pedido tem de estar sempre associado a uma variável. Vamos então primeiro criar a variável.

Com o botão direito, clicar sobre o menu *Variables (Tags)* e dar o seu nome como exemplificado na Figura 6.24.

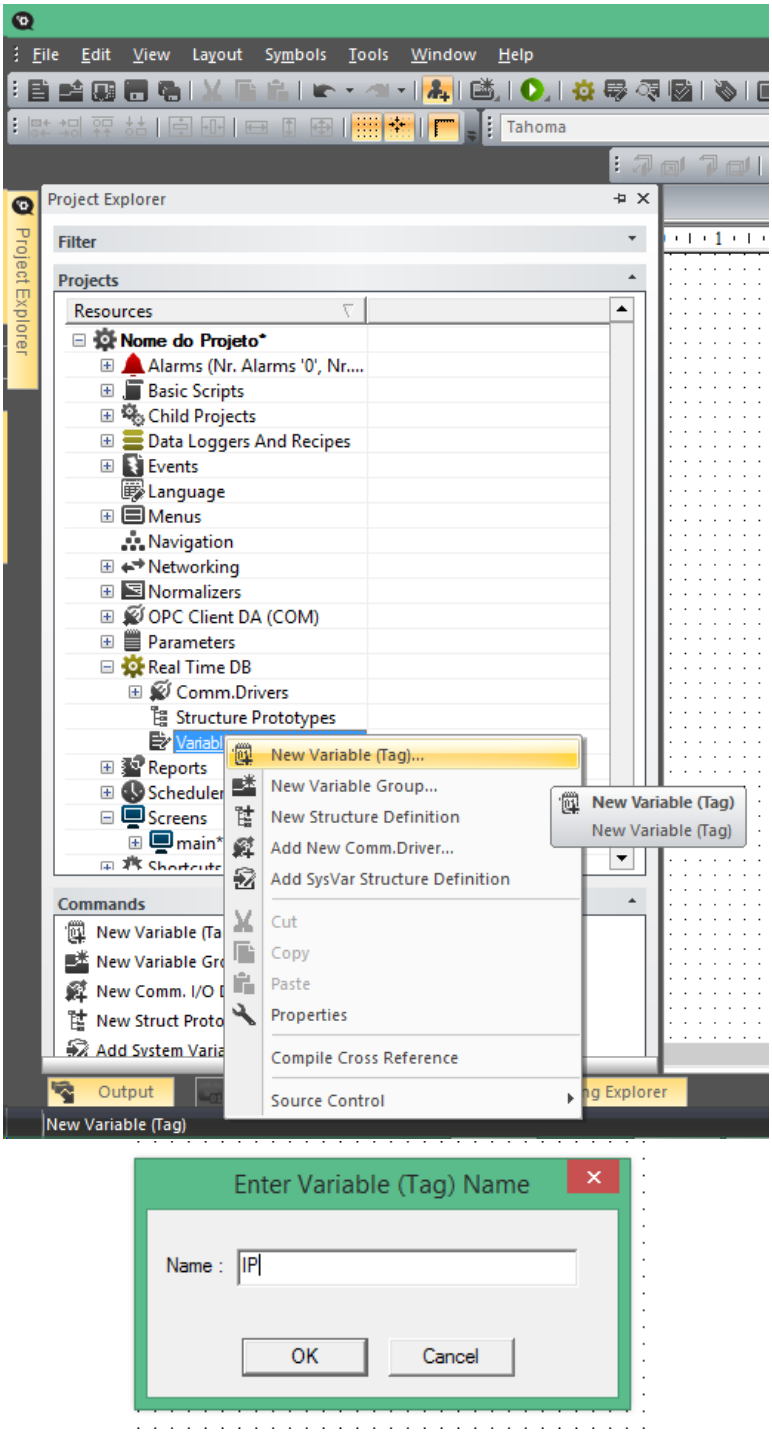


Figura 6.24: Criação de uma variável no software Movicon

Depois, clicar com o botão direito na variável, selecionar o menu *Properties* e abrir a janela *Dynamic* como mostra a Figura 6.25.

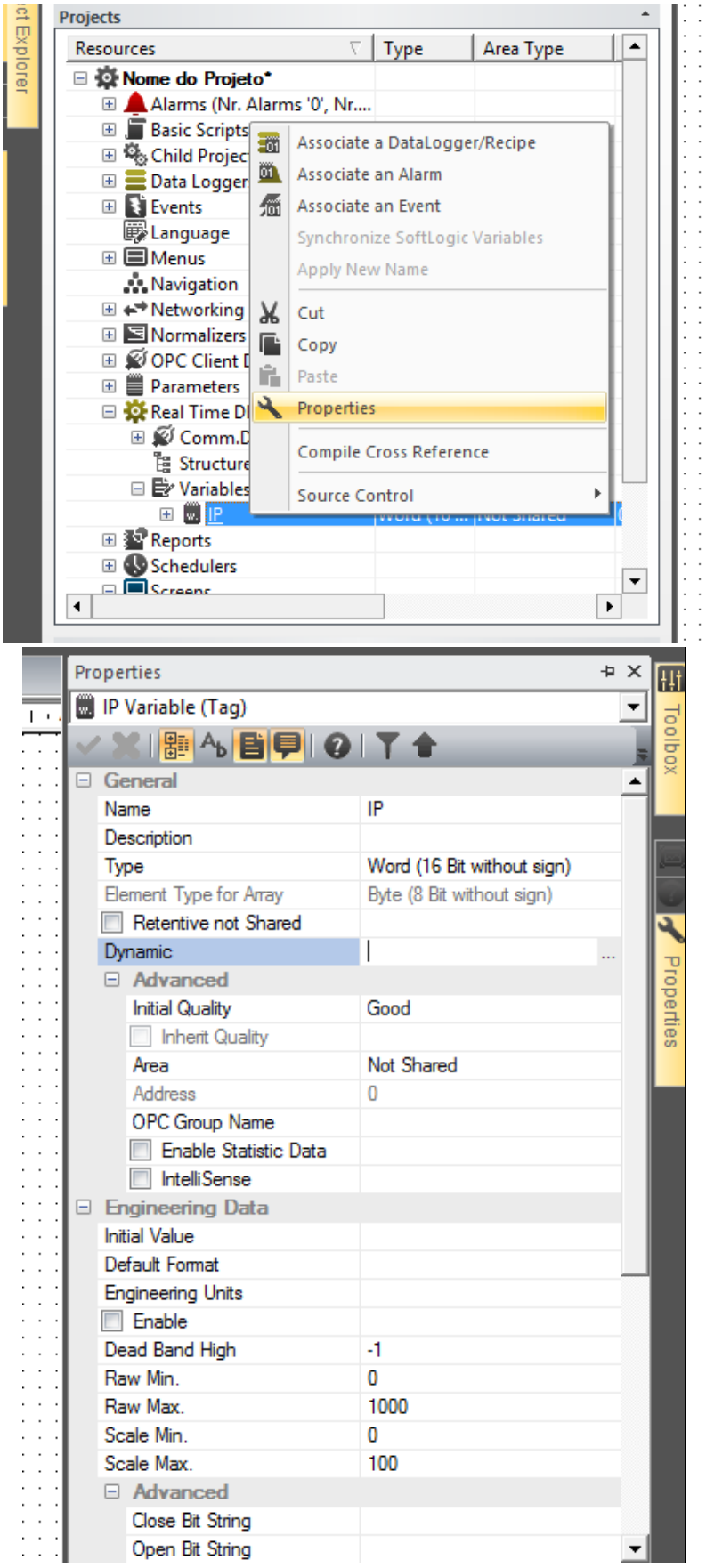


Figura 6.25: Alteração de propriedades de uma variável no software Movicon

Agora vamos configurar o pedido relativo à variável em questão. De notar que como o pedido de IP se trata de um número e que vem com informação de 4 bytes (8 caracteres em Modbus RTU), então a variável é do tipo *DWord (32 bit without sign).*

Depois de aberta a janela *Dynamic* relativa à variável, vamos clicar no botão *Add/Edit* indicado na Figura 6.26.

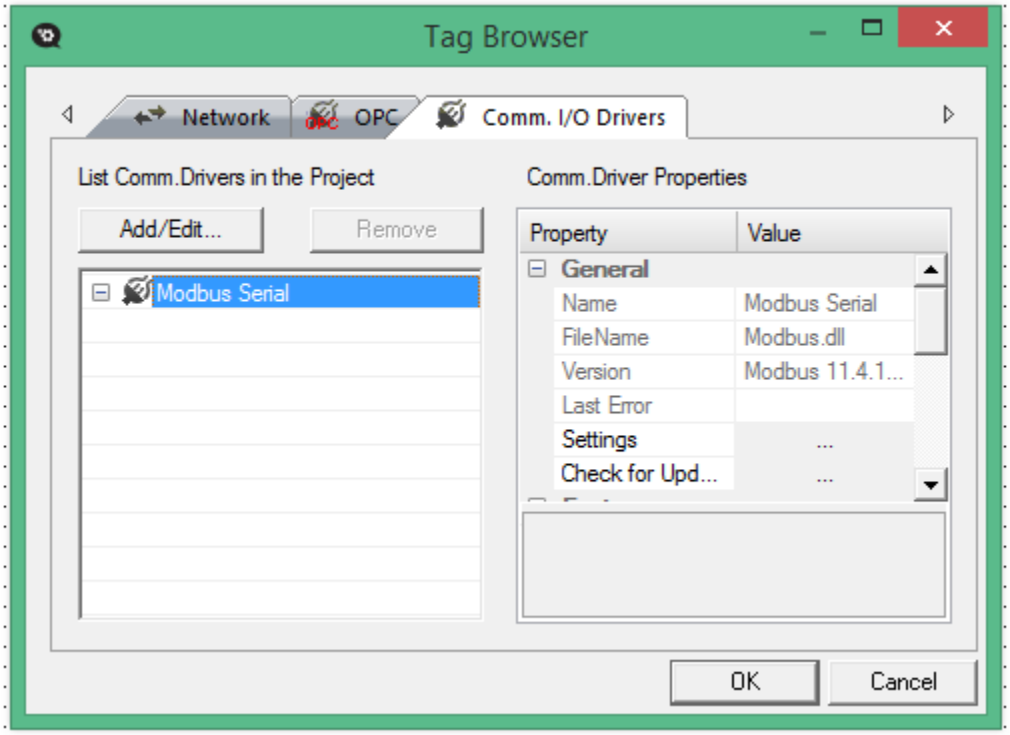


Figura 6.26: Menu para criação/alteração de uma ligação no software Movicon

E irá aparecer a janela de configuração do pedido *Modbus RTU* representado na Figura 6.27.

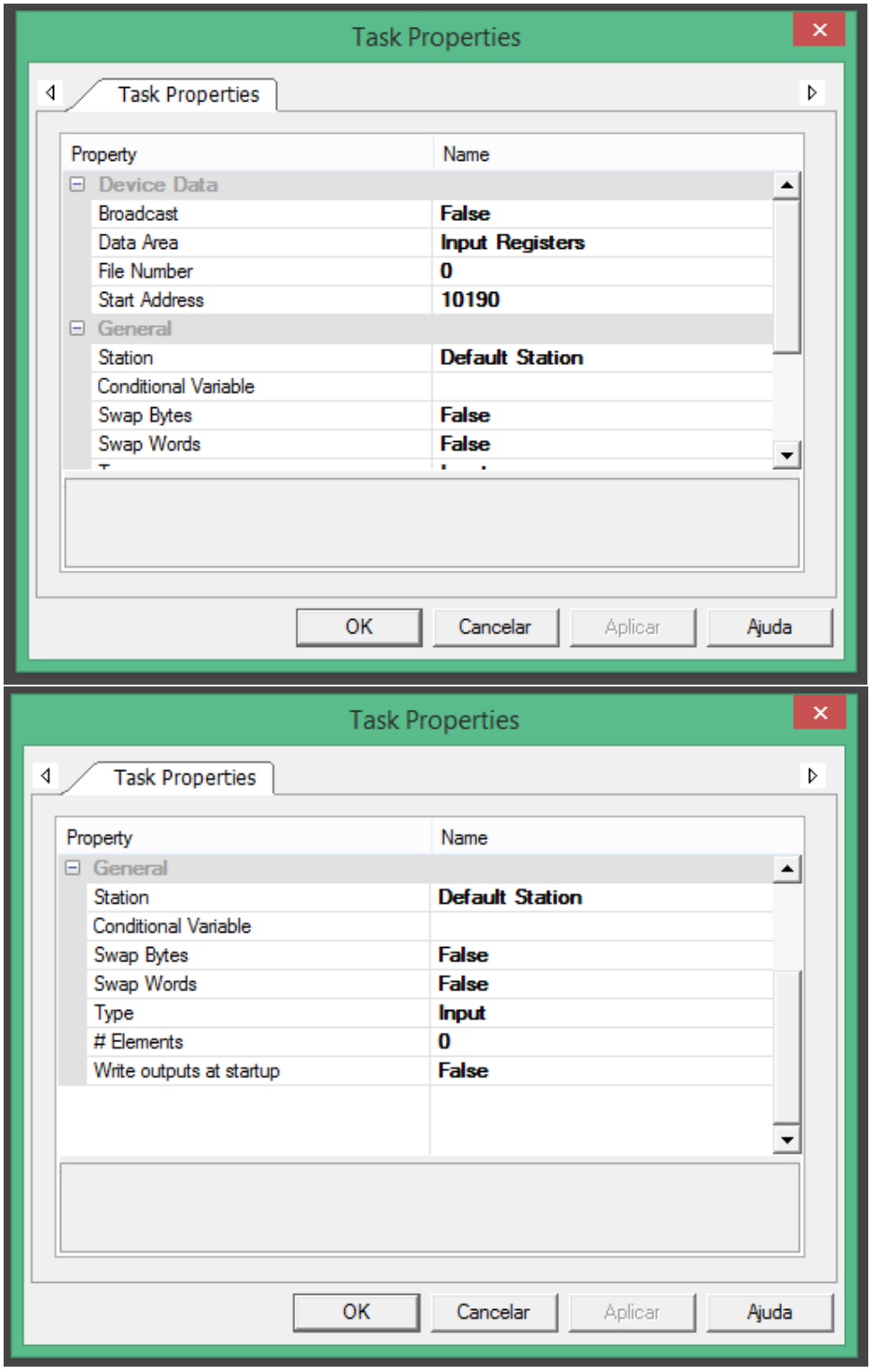


Figura 6.27: Configuração de uma ligação Modbus RTU *no* software Movicon

Agora com as ligações e o pedido configurado, podemos construir a interface gráfica. Para isso vamos adicionar uma nova janela e dar o nome. Este passo está representado na Figura 6.28.

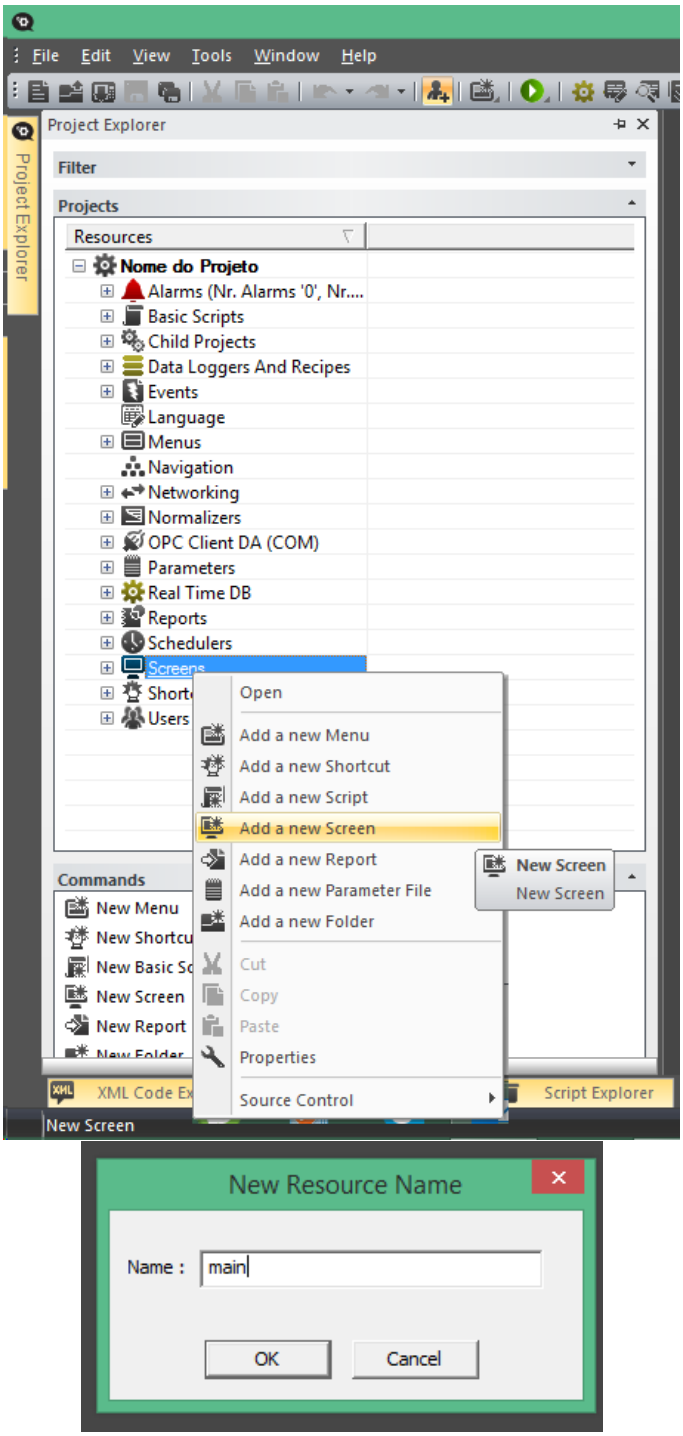


Figura 6.28: Adição de um novo ecrã no software Movicon

Com a *toolbox* apresentada na Figura 6.29 podemos escolher os elementos que queremos na interface gráfica e fazer *drag and drop*.

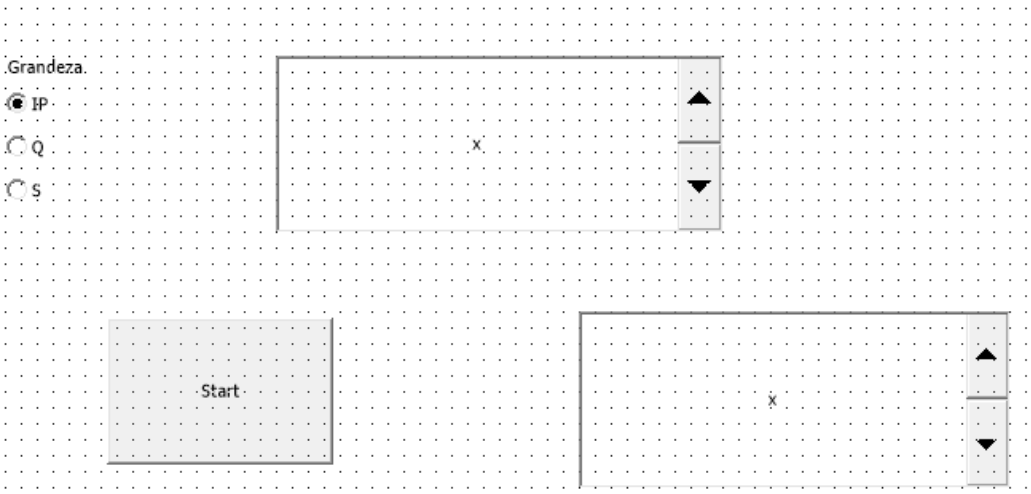


Figura 6.29: adição de elementos a um ecrã no software Movicon

As *Edit box Display* podem mostrar valores de variáveis, para isso basta fazer *drag and drop* da variável para dentro da caixa.

No caso dos *radio buttons*, para podermos configurar o título do conjunto e os nomes das várias opções temos de colocar o texto na propriedade *Object Title*, como é apresentado na Figura 6.30. Para associarmos uma variável, temos de criar uma à imagem de como fizemos anteriormente e adicionar como mostra a figura seguinte, no campo *Command/State Variable.* Para definirmos o número de opções, é no parâmetro *Radio buttons number*.

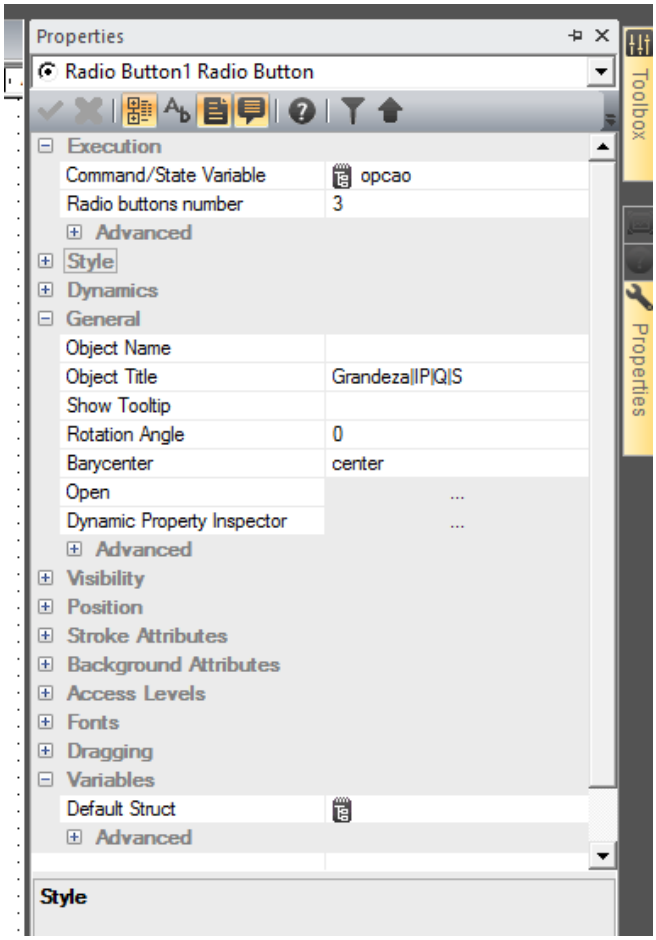


Figura 6.30: Propriedades do elemento Radio Button *no* software Movicon

Nota: neste caso temos três opções, então a variável opção vai tomar os valores 0,1 ou 2, consoante a opção selecionada, sendo a opção 0 a primeira no sentido descendente. Esta informação pode ser útil por exemplo num *Script.*

Quanto ao botão *Start*, podemos editar o seu texto, na propriedade *Object Title* como mostra a Figura 6.31

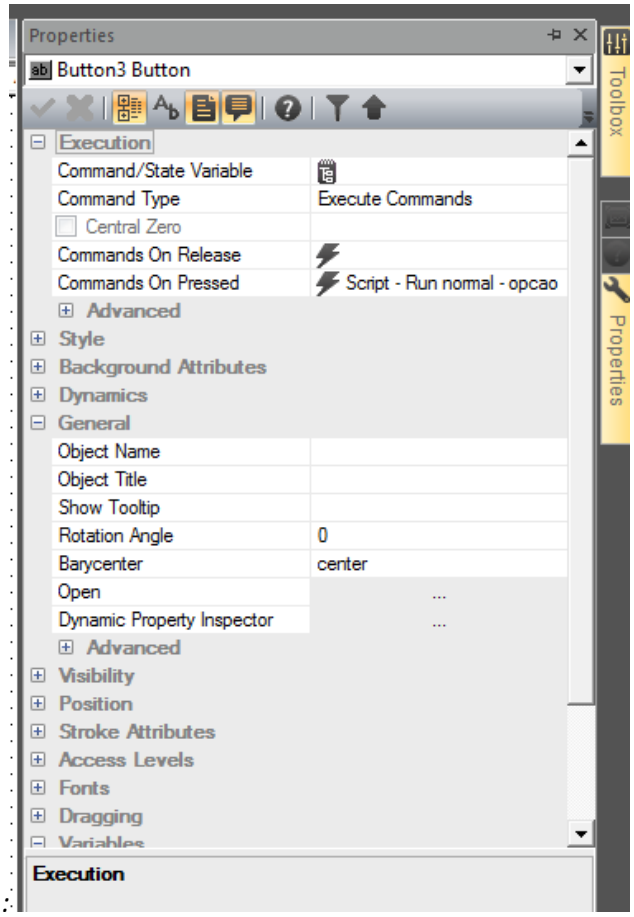


Figura 6.31: Propriedades do elemento Button *no* software Movicon

Mudando o tipo na propriedade *Command Type*, podemos fazer com que este botão, quando primido corra um *Script.*

Vamos agora ver o *Script*, que foi escrito para mostrar o IP no formato 193.137.172.23

Para criar um *Script*, temos de fazer de forma muito semelhante à criação de variáveis e novos ecrãs, como mostra a Figura 6.32.

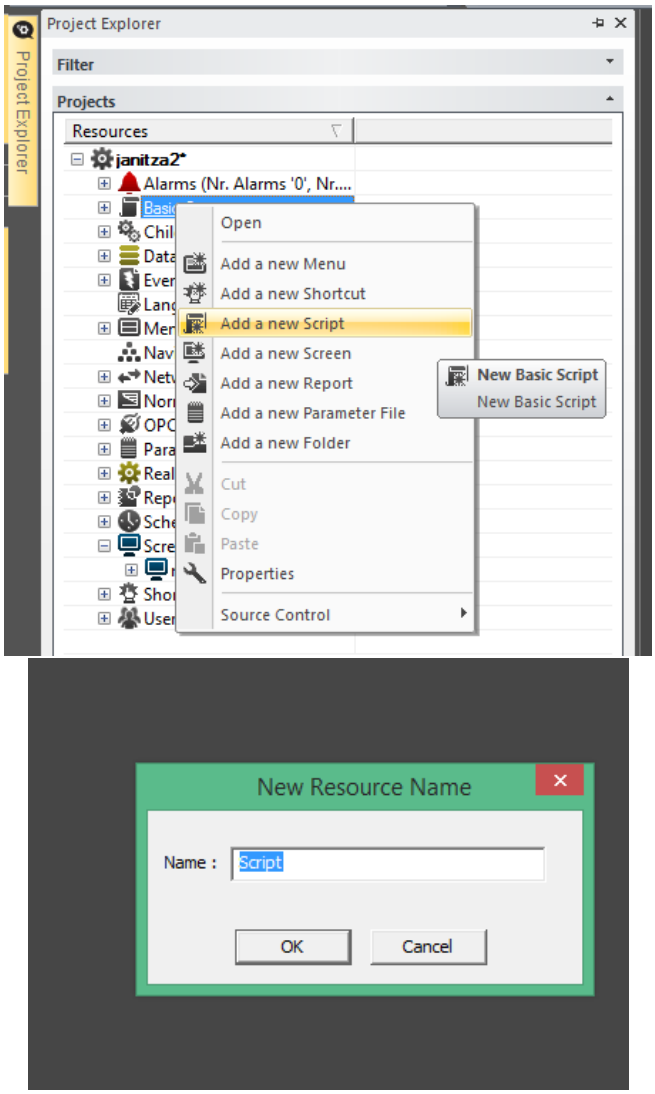


Figura 6.32: Adição de um script *no* software Movicon

Dado o nome ao *Script*, temos de o escrever. O *Script* utilizado neste teste é o seguinte:

E.1.15 Script

```
'#Language "WWB.NET"

	Sub Main()

	If opcao=0 Then
		Dim NumHex As String
		NumHex = Hex(ip)

		Dim aux1 As String
		aux1=""

		Dim aux2 As String
		aux2=""

		Dim index As Integer

		For index = Len(NumHex) To 2 Step -2
			aux1 = aux1 & WWB.GetChar(NumHex, index-1) &
WWB.GetChar(NumHex, index)
		Next

		For index =2 To Len(aux1) Step 2
			aux2 = aux2 & Val("&H" & WWB.GetChar(aux1,index-1) &
WWB.GetChar(aux1,index))
			If index<Len(aux1) Then
				aux2=aux2 & "."
			End If

		Next

		opcao_text="IP " & aux2

	End If

	If opcao=1 Then
		opcao_text="Reactive Power"
	End If

	If opcao=2 Then
		opcao_text="Total Power"
	End If


End Sub
```

E.1.16 Teste

No teste realizado, a opção *Swap Bytes* e *Swap Words*, da figura 1.15, foram colocadas a *False*. Assim no *script* acima referido, teve de ser feita a inversão dos *Bytes*.

O resultado foi o que mostra a Figura 6.33.

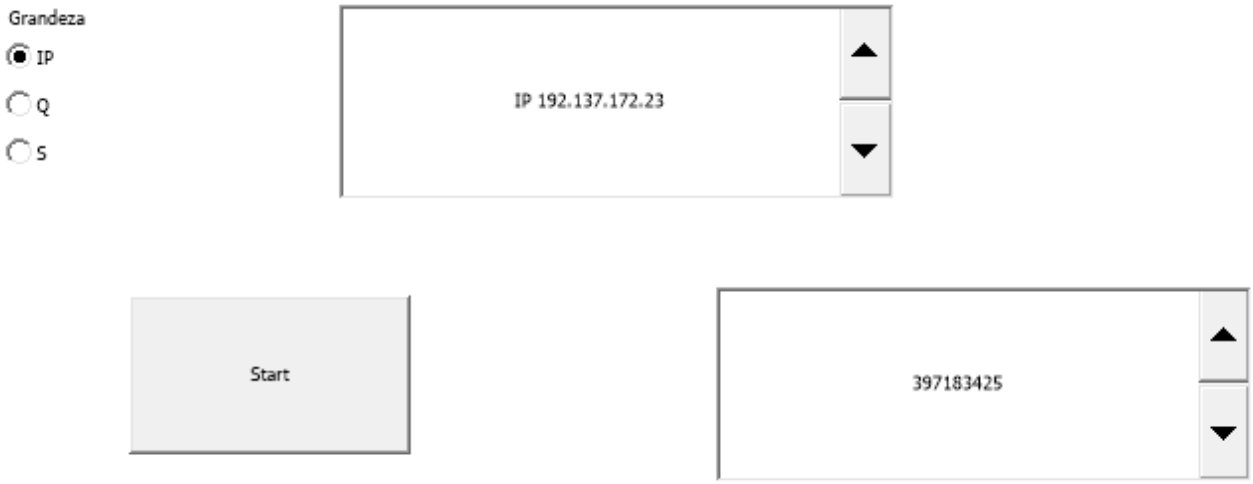


Figura 6.33: Resultado da execução do programa Movicon

Na caixa de texto mais à direita, temos o valor da variável associada ao pedido *Modbus*. Na caixa de texto mais à esquerda, temos o resultado depois de aplicado o *script*.

Anexo F

F.1 Rede de contadores de energia elétrica

F.1.1 Introdução

Na empresa Colep Vale de Cambra, existem contadores de energia com vários tipos de comunicação e para os mais variados tipos de energia. No caso em questão, o interesse cai sobre os contadores de energia elétrica. Estes contadores são na maioria possuidores de uma **saída de impulso**, que surge n vezes (depende da marca e modelo) quando o contador conta mais uma unidade (1 kWh por exemplo). Outros contadores têm comunicação por ***RS485*** sob o protocolo ***Modbus RTU***.

Nesta empresa, existe também uma base de dados em ***Microsoft Access*** que contém os valores **mensais** das contagens. Esta base de dados é utilizada para fazer cálculos de contagem por secções da fábrica. Além disso serve para fazer um histórico de consumos nessas mesmas secções ao longo dos anos.

Com este trabalho é pretendido fazer uma monitorização em tempo real utilizando uma base dados idêntica à existente mas com um tempo de amostragem mais curto (por exemplo uma amostra a cada minuto). O esquema do sistema pretendido é representado na Figura 6.34

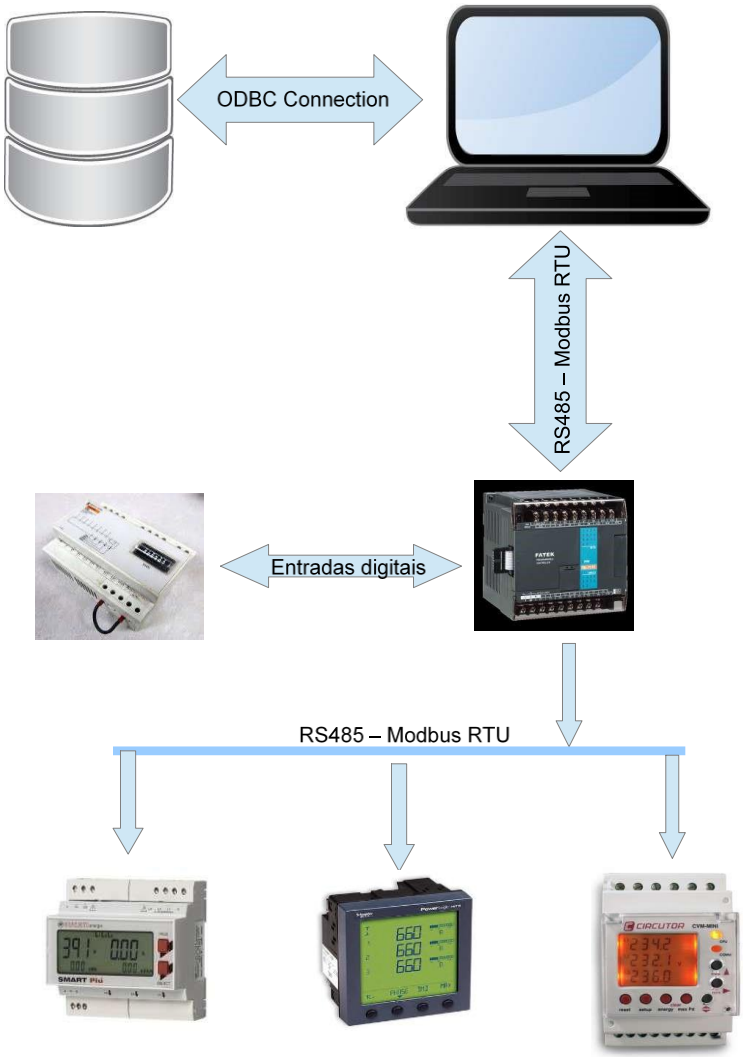


Figura 6.34: Arquitetura pretendida

F.1.2 Objetivos

- Fazer uma rede de leitura dos valores dos vários tipos de contadores em tempo real
- Atualizar em tempo real a base de dados

F.1.3 Resumo

Para realizar este trabalho, foi pensada uma rede de várias camadas. A **camada de alto nível** onde temos o computador que comunica com a base de dados por linguagem **SQL** através de uma ligação **ODBC**. Uma **camada de nível médio** onde a informação vinda dos contadores é tratada e armazenada de maneira a ser sensível ao computador. E por fim, **uma camada de baixo nível** onde temos os contadores da fábrica.

F.1.4 Camada de alto nível

É onde se encontra o computador ligado a uma base de dados. A ligação é do tipo *ODBC* que foi escolhida pela familiarização com esta e pela facilidade em ambiente *Windows/Visual Basic*. Para o computador se conectar à rede e recolher os valores para os enviar para a base de dados, foi desenvolvida uma aplicação em ***Visual Basic,*** pois é uma ferramenta que permite o

desenvolvimento muito rápido e de forma fácil de aplicações para *Windows* com interfaces agradáveis. Além disso a complexidade exigida para esta aplicação é perfeitamente suportada por esta linguagem. Na Figura 6.35 está representada uma figura ilustrativa desta camada.

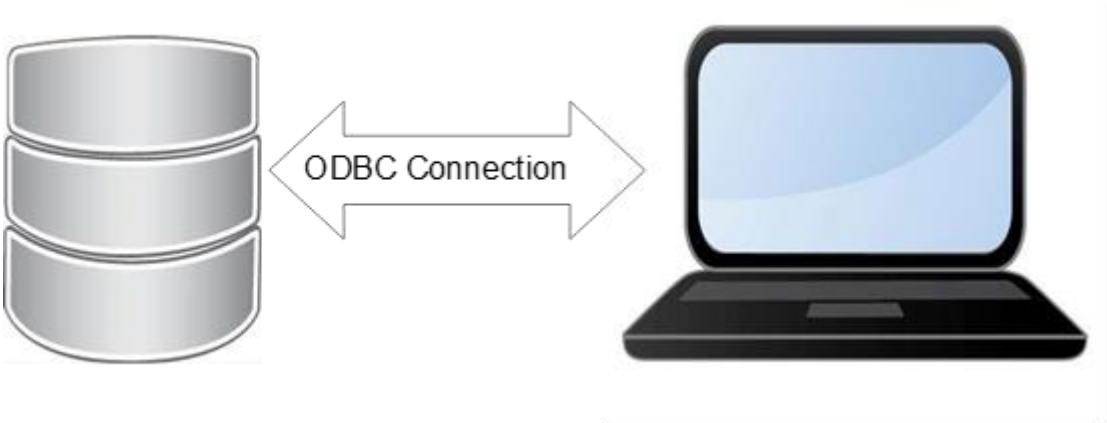


Figura 6.35: Camada de alto nível

F.1.5 Camada de nível médio

Aqui é onde os dados/sinais vindos do "chão de fábrica" são tratados. A opção escolhida foi um conjunto de autómatos, pois o custo da marca em causa (Fatek) é competitivo quando comparado com aparelhos dedicados apenas para este fim como o caso dos *remote IO's.* Assim por um preço baixo temos um aparelho mais versátil. Esta camada é de extrema importância pois é aqui que os diversos tipos de informação, sejam em *RS485 ModBus RTU*, seja impulso elétricos, se vão tornar num único tipo de informação, que é sensível ao computador. Na Figura 6.36 está ilustrada a arquitetura desta camada.

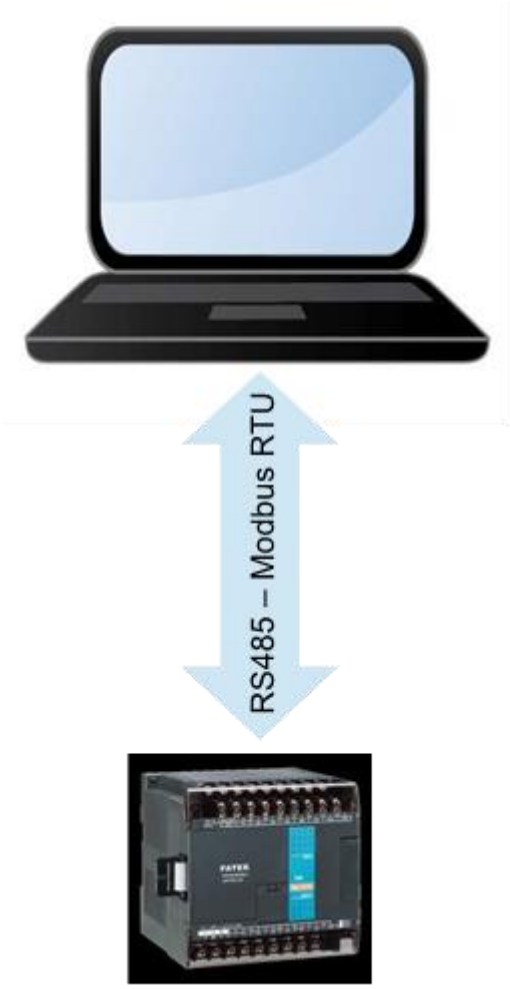


Figura 6.36: Camada de médio nível

F.1.6 Camada de baixo nível

Por último, esta camada é onde se encontram os contadores dos mais variados tipos e marcas. Esta camada é diversificada e é a partir daqui que temos de pensar o trabalho. A representação da camada está ilustrada na Figura 6.37.

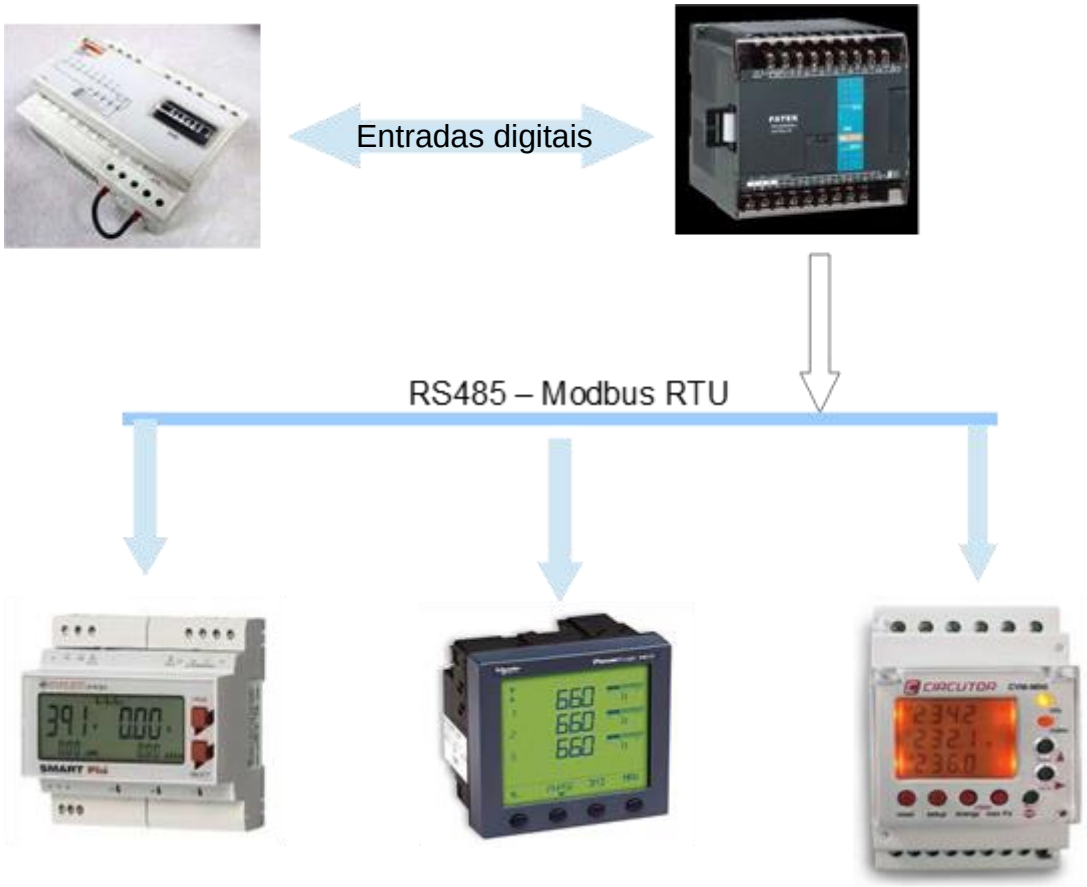


Figura 6.37: Baixo nível

F.1.7 Rede de teste

Antes da rede de contadores ser colocada a funcionar, irá ser feita uma rede de teste. Para isso vamos utilizar um autómato da marca Fatek, apenas um contador com comunicação *RS485 ModBus RTU* da marca *Janitza* e um botão de pressão para simular um contador de impulso. O autómato servirá como acumulador de impulsos e valores de contagens. Será também responsável por comunicar com o contador que possui comunicação e irá guardar os valores em posições de memória específicas. Quanto ao contador gerador de impulsos (simulado pelo botão) será encarado pelo autómato como entrada digital e este apenas fará a contabilização do nº de impulsos tendo em conta o *offset* do contador que lhe será comunicado pela aplicação em *Visual Basic,* que estará a correr no computador. O diagrama da arquitetura deste teste está representado na figura Figura 6.38.

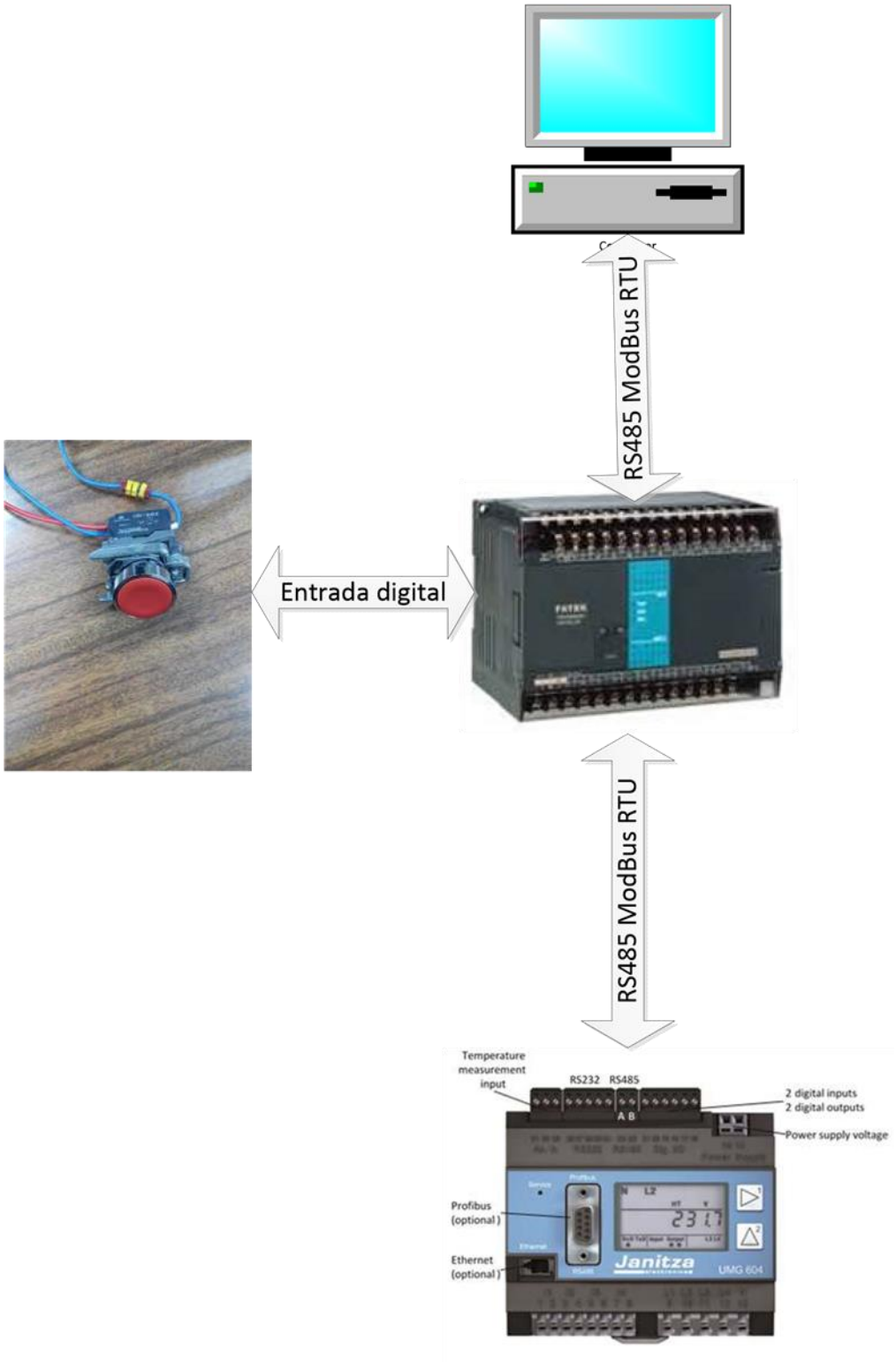


Figura 6.38: Rede de teste

F.1.7 Material utilizado na rede de teste

As imagens seguintes representam os vários aparelhos utilizados para realizar o teste.

- Fatek FBs-20MC Figura 6.39

Figura 6.39 Imagem ilustrativa do autómato utilizado [35]

- FBs-CM25E Figura 6.40Figura 6.40 Imagem Ilustrativa da expansão utilizado com o autómato

Figura 6.40 Imagem Ilustrativa da expansão utilizado com o autómato [36][36]

- Janitza UMG 604 Figura 6.41

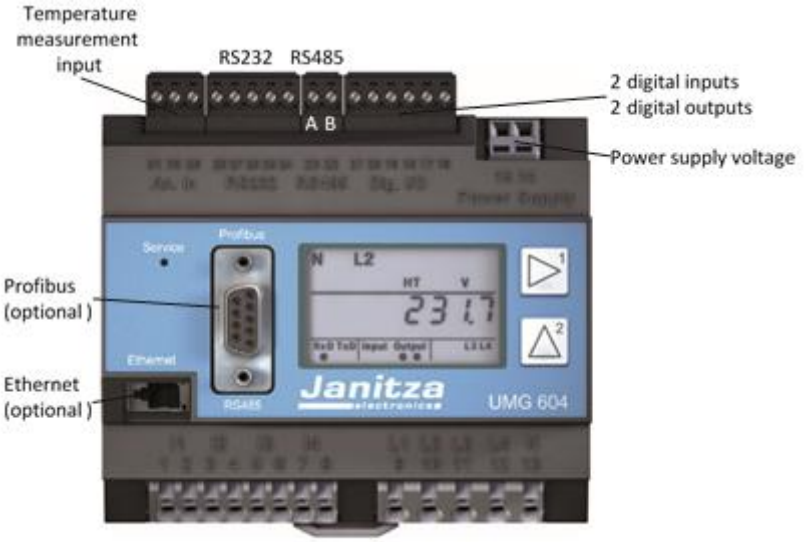


Figura 6.41Figura 6.42 Aparelho Janitza UMG604 [37]

- Schneider Electric ZBE – 101 Figura 6.43

Figura 6.43: Botão de pressão

- Botão de pressão Figura 6.44

Figura 6.44: Botoneira

F.1.8 Norma ModBus RTU

Primeiro que perceber a arquitetura da solução proposta é necessário perceber o protocolo utilizado, o protocolo *ModBus RTU*. Este protocolo impõe uma estrutura às mensagens trocadas entre dois ou mais aparelhos constituintes de uma rede. Um desses aparelhos terá de ser o *master*, e todos os outros aparelhos serão os *slaves*. O *master* é o aparelho responsável por realizar pedidos aos *slaves* e estes apenas respondem aos pedidos que lhes são feitos. Como o protocolo apenas define a estrutura da mensagem, este pode ser utilizado em vários meios de transmissão físicos, mas os mais usados são as comunicações série, RS232 e RS485. À semelhança deste protocolo, existe o *ModBus Ascii*, que em tudo é semelhante ao *RTU* tendo apenas algumas pequenas diferenças. O mais utilizado por aparelhos industriais é o *RTU*.

F.1.9 Estrutura

A estrutura de uma mensagem *ModBus RTU* é composta por vários campos como indicado na Tabela 6.7.

Tabela 6.7: Estrutura de mensagem Modbus RTU

Endereço	Função	Data	CRC
1 Byte	1 Byte	N x 1 Byte	2 Byte

Nota: 1 byte = 8 bits

F.1.10 Endereço:

Cada aparelho na rede possui um número de 1 Byte que é a sua identificação na rede. Este número não pode ser repetido pois é o "Endereço" de cada aparelho. Para qualquer comunicação direcionada a um aparelho específico é preciso o seu endereço.

F.1.11 Função:

O *ModBus* disponibiliza uma série de funções que o master pode pedir para o *slave* realizar, entre as quais, ler um registo, forçar o valor de uma saída digital, etc. Neste campo é onde é colocado o código da função que é codificada em 8 bits.

F.1.12 Funções existentes na norma ModBus

01 READ COIL STATUS (Lê estado de memória booleanas.
Nota: 1 memória booleana = 1 bit

02 READ INPUT STATUS (Lê o valor de entradas)

03 READ HOLDING REGISTERS (Lê o valor de memórias retentivas)

04 READ INPUT REGISTERS (Lê o valor de registos de entrada)
Nota: 1 registo = 16 bits, 2 bytes)

05 WRITE SINGLE COIL (Edita o estado de uma memória booleana)

06 WRITE SINGLE REGISTER (Edita o valor de 1 registo)

15 WRITE MULTIPLE COILS (Edita o estado de várias memórias booleanas)

16 WRITE MULTIPLE REGISTERS (Edita o valor de vários registos)

F.1.13 Data:

Aqui é onde é escrita a informação necessária para realizar as mais variadas funções como por exemplo a posição de memória a ler e o número de posições consecutivas. Este campo depende da função e se se trata de um pedido ou resposta.

F.1.14 CRC

Conhecido como CRC o "*cyclic redundancy check*" é um número de 2 Bytes que resulta de um algoritmo matemático que utiliza os restantes Bytes da mensagem. Desta forma o CRC é um código que pode ser utilizado para verificar se a informação enviada é a mesma que a recebida.

Nota: Normalmente, as representações de mensagem *ModBus RTU* são feitas com caracteres hexadecimais, pois todos os campos têm tamanhos múltiplos da unidade *Byte* e **um caracter hexadecimal precisa de 4 bits para ser codificado**. Assim, ao longo do documento **todos os campos de mensagens *ModBus RTU* vão estar representados por um número múltiplo de 2 caracteres hexadecimais (2caracteres = 8bits = 1Byte)**.

Para percebermos melhor o significado destes campos, vamos ver um exemplo de uma transação específica:

F.1.15 Pedido

Na Tabela 6.8 está representado um pedido ModBus RTU.

Tabela 6.8: Pedido enviado

Endereço do *slave* (1 Byte)	Função (1 Byte)	Posição de começo (2 Bytes)	Quantidade de endereços (2 Bytes)	CRC (2 Bytes)
04	01	00 0A	00 0D	DD 98

Endereço do *slave:* Endereço do aparelho a que o pedido se destina. Neste caso é o aparelho número 4.

Função: Código da função, neste caso trata-se de uma função para ler o estado de memórias com apenas um bit, conhecidas como *Coils*. A função com o código número 1 na norma *ModBus* tem a o nome de *"Read Coils"*

Posição de começo: Endereço da posição de memória onde é pretendido fazer a leitura. Neste caso quer-se ler a partir da posição 10 inclusive (A em hexadecimal).

Quantidade de endereços: Número de posições de memória que é desejado ler, que são 13 (D em hexadecimal)

CRC: Verificação CRC.

F.1.16 Resposta

A resposta ao pedido efetuado acima está representado na Tabela 6.9.

Tabela 6.9: Resposta ao pedido

Endereço do *slave* (1 Byte)	Função (1 Byte)	Contagem de Bytes (1 Bytes)	Informação dos endereços (2 Bytes)	CRC (2 Bytes)
04	**01**	**02**	**0A 11**	**B3 50**

Endereço do *slave:* Endereço do aparelho a que pertence a mensagem de resposta.

Função: Código da função a que o *slave* está a responder.

Contagem de Bytes: Número de Bytes que o campo **Informação dos Endereços** contém.

Informação dos Endereços: Informação pedida pelo master. 0A 11_{16} = 0101000010001_2. Cada bit da resposta representa o estado de cada *coil*, sendo o bit da esquerda o referente à posição de memória mais significativa.

CRC: Verificação CRC.

F.1.17 Comunicação entre camada de nível médio e baixo nível por RS485 ModBus RTU *(entre autómato e o contador)*

A comunicação entre estes 2 dispositivos é feita através de *RS485* segundo o protocolo *Modbus RTU.* O autómato será o *Master* da ligação e o analisador de energia será o *Slave*. Neste teste vamos realizar 2 pedidos ao aparelho. O primeiro da posição 9851, que é "*Real Energy, Supply, L1+L2+L3"* medido em *Wh*. O segundo pedido é da posição 19050, que é "*Measured frequency*" em *Hz*. A representação esquemática está representada na Figura 6.45.

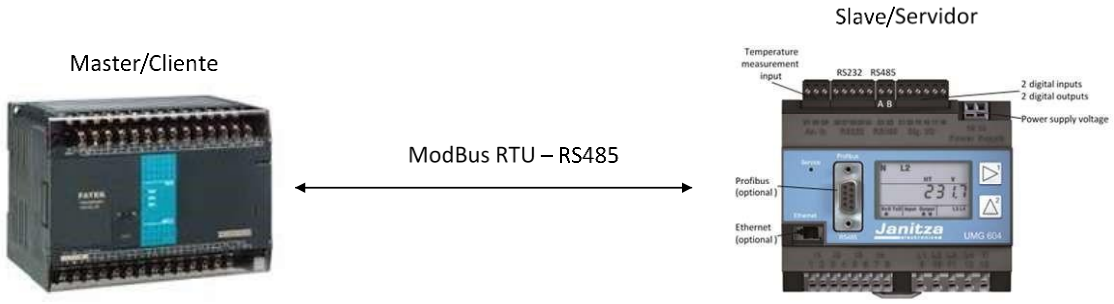


Figura 6.45: Representação da ligação

F.1.18 Pedido "Real Energy, Supply"

O analisador *Janitza* foi definido com o endereço 1. A função escolhida foi a função 3, *Read Holding Registers*, mas a função 4 também funciona para este caso. O campo do endereço terá os seguintes Bytes, sendo o 1º o mais significativo: 26 7B ($267B_{16}$ = 9851_{10}). Estes valores podem ser consultados no manual de endereços *Modbus*, do aparelho *Janitza.* O número de registos a ler serão 2, pois cada registo representa 1 *word*, que são 16 bits e o campo em causa tem um valor que segundo o manual de endereços do aparelho *Janitza* é um *float* de 32 bits. O *CRC* pode ser calculado num calculador *online* por exemplo. Neste caso foi utilizado uma ferramenta para o efeito presente no programa *Winproladder* que serve para programar os autómatos da marca *Fatek*.

A palavra que resulta pode ser vista na Tabela 6.10:

Tabela 6.10: Pedido "Real Energy, Supply"

Endereço do *slave* (1 Byte)	Função (1 Byte)	Posição de começo (2 Bytes)	Quantidade de endereços (2 Bytes)	CRC (2 Bytes)
01	**03**	**26 7B**	**00 02**	**BF 5A**

A transação está representada na Figura 6.46.

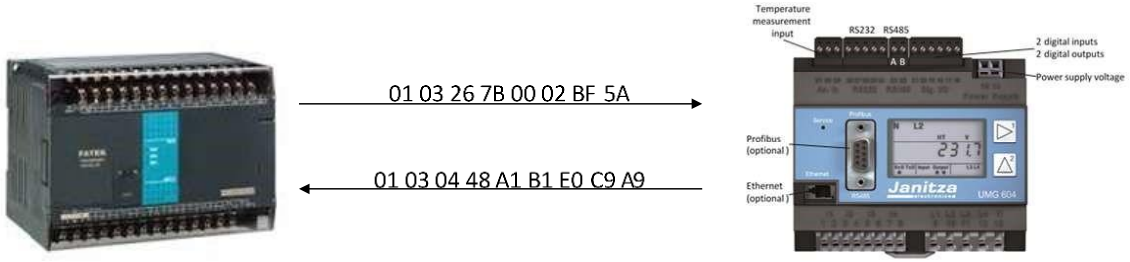


Figura 6.46: Representação da comunicação

A resposta do por parte do *slave* foi a seguinte, representada na Tabela 6.11:

Tabela 6.11: Resposta do Slave

Endereço do *slave* (1 Byte)	Função (1 Byte)	Contagem de Bytes (1 Bytes)	Informação dos endereços (4 Bytes)	CRC (2 Bytes)
01	03	04	48 A1 B1 E0	C9 A9

O número $48A1B1E0_{16} = 1218556384_{10} = 0100\ 1000\ 1010\ 0001\ 1011\ 0001\ 1110\ 0000_2$. No entanto esta não é a resposta final. Este número está no formato *IEEE-754 Floating-Point* segundo o manual de endereços do aparelho *Janitza*. Para realizar a conversão é necessário saber que um número no formato *IEEE-754 Floating-Point* é representado em decimal da seguinte forma que está representada na Equação 4.3.1.1-1:

Equação 4.3.1.1-1: representação de um número IEEE-754 em decimal

$$S\ M \times 2^E$$

Sendo ***S*** conhecido como **sinal,** trata-se **do bit mais significativo** (bit 31). Este bit a 0 significa + e o bit a 1 significa -. Neste caso o bit 31 é **0.**

M* é mantissa** ou parte fracionária do número 1 e corresponde aos valores do **bit 0 ao 22** inclusive. O valor de ***M é: 1.**010 0001 1011 0001 1110 0000**$_2$

***E* é o expoente** e é representado pelos **bits da posição 23 ao 30** inclusive subtraído de 127_{10}. Então o valor que resulta é: $100\ 1000\ 1_2$ - 1111111_2

Assim sendo, a resposta é igual a:

$$+ 1.010\ 0001\ 1011\ 0001\ 1110\ 0000\ \times\ 2^{1001\ 0001 - 111\ 1111}$$

$$\Leftrightarrow$$

$$1.2632408 \times 2^{18} = 331151.00\ Wh \sim \mathbf{331.2kWh}$$

O resultado está certo como podemos verificar na Figura 6.47:

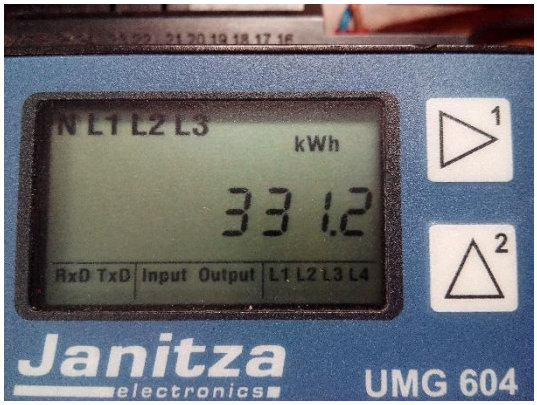


Figura 6.47: Valor no aparelho Janitza

F.1.19 Pedido "Measured Frequency"

O endereço do aparelho e a função utilizada são os mesmos do exemplo anterior. Quanto à posição de memória é agora $19050_{10} = 4A\ 6A_{16}$. Mais uma vez, trata-se de um *float* de 32 bits, por isso o número de registos a ser lidos são 2. O *CRC* mais uma vez é calculado com recurso a um calculador.

A palavra que resulta pode ser vista na Tabela 6.12:

Tabela 6.12: Pedido "Measured Frequency"

Endereço do *slave* (1 Byte)	Função (1 Byte)	Posição de começo (2 Bytes)	Quantidade de endereços (2 Bytes)	CRC (2 Bytes)
01	03	4A 6A	00 02	F2 0F

Realizando o pedido, a transação seguinte, que está representada na Figura 6.48, é efetuada:

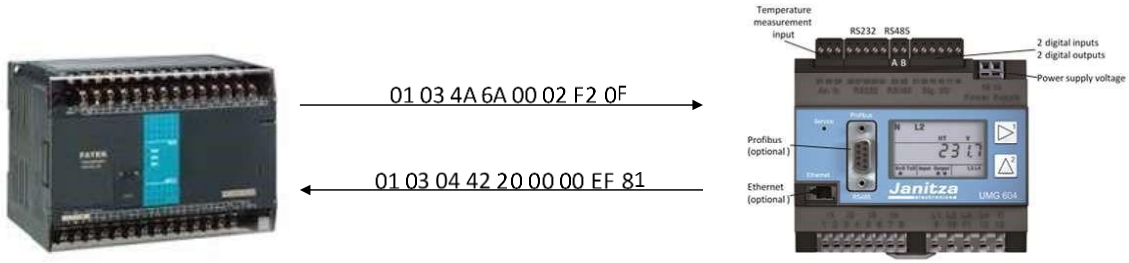


Figura 6.48: Representação da comunicação

A resposta por parte do *slave* foi a seguinte representada na Tabela 6.13:

Tabela 6.13: Resposta ao pedido "Measured Frequency"

Endereço do *slave* (1 Byte)	Função (1 Byte)	Contagem de Bytes (1 Bytes)	Informação dos endereços (4 Bytes)	CRC (2 Bytes)
01	03	04	42 20 00 00	EF 81

À imagem do exemplo anterior, este registo também está codificado no formato *IEEE-754 Floating-Point*. Seguindo o mesmo raciocínio temos o valor resultante: 40.000000 Hz

O resultado está certo como podemos verificar na Figura 6.49:

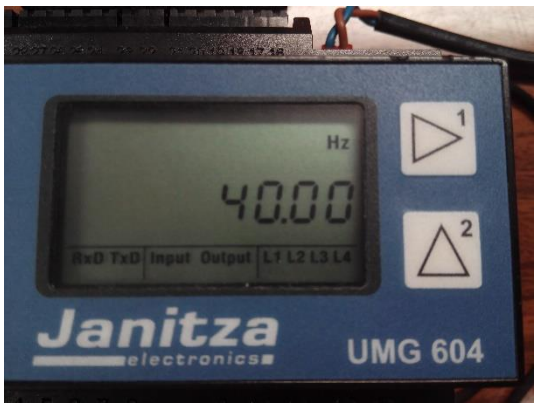


Figura 6.49: Valor no aparelho Janitza

F.1.20 Implementação do protocolo Modbus RTU master na interface PORT4 (RS485)

Na secção anterior foi explicada a transação entre o autómato e o analisador de energia. Agora vai ser explicado como programar o autómato para que este realize pedidos *Modbus RTU* por uma porta específica.

Tão ou mais importante do que implementar um protocolo é a configuração da ligação numa porta série. Assim antes de definir o protocolo, irão ser definidos os parâmetros da ligação. Os parâmetros escolhidos foram os seguintes: **9600 bps, 8 data bits, No Parity, 1 stop bit.**

Para definir estes parâmetros é necessária a consulta do **capítulo 12 do manual do autómato , secção 12.4.3** [30]**.** Da consulta obtêm-se a seguinte informação:

R4044: -1º byte=56H

-2º byte=0 (Even Parity [é indiferente]) + 1 (8 data bits) + 0 (None Parity) + 1 (para 1 stop bit) + 0001 (9600 bps)

Então é necessário colocar o valor **5641H no registo R4044**, para aplicar os parâmetros requeridos.

Aplicados os parâmetros da comunicação, vai-se agora implementar o protocolo. Basta para isso utilizar a função **150.M-Bus**, ilustrada na Figura 6.50 e explicada a partir da **página 40 do capítulo 13 do manual** [38], para assim enviar mensagens *Modbus RTU*, e receber a hipotética resposta.

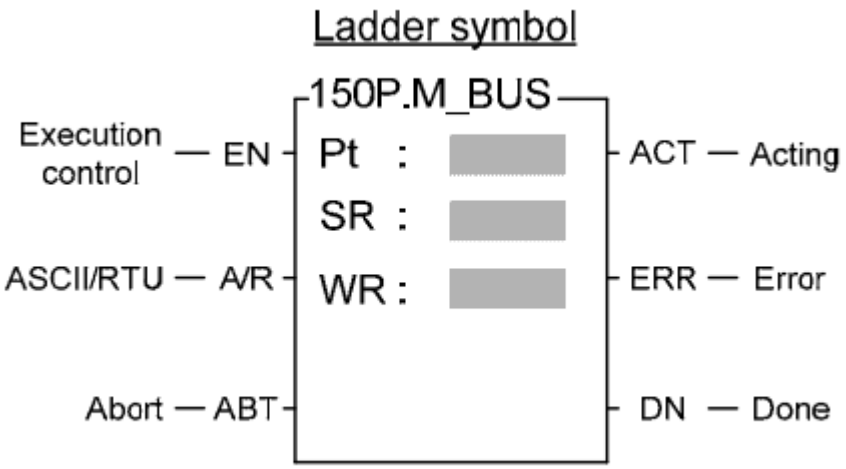


Figura 6.50 Imagem Ilustrativa da função 150.M-Bus (adaptado de [38]*)*

Ao utilizar esta função é necessário definir 3 valores. ***Pt*** é a porta que irá transmitir a informação. Neste caso é a 4. ***SR*** é o registo inicial onde é colocada a informação que vai conter a mensagem *Modbus*. E ***WR*** é registo inicial onde irá ser escrita a informação de como decorreu a ação (a ser preenchido pelo autómato).

Na **página 45 do capítulo 13** [38], está explicado o significado de cada registo **SR**, e que é detalhado a seguir:

Definindo o *SR* como sendo *R0*, por exemplo, tem de se colocar o valor **0** no campo **SR** e colocar os seguintes valores nos registos de **R0** em diante:

- **SR+0=R0: A550H** (valor predefinido a colocar no 1º registo)
- **SR+1=R1: 1º BYTE=07H, 2BYTE=nº** de mensagens *Modbus* a enviar. Isto é: realizando um pedido com a função 04 e fazendo o *set* de uma saída com a função 05, ou então realizando o pedido do valor de 2 posições de memória não consecutivas por exemplo, este campo tem de ter o valor **2**.

A partir deste registo, vamos escrever em **7xN registos sendo N o 2byte do registo R1**. Ou seja, cada mensagem precisa e 7 registos para ser configurada.

- **SR+2=R2**: nº do *slave*
- **SR+3=R3**: código do comando a realizar. Atenção! Este código não é o mesmo que é definido pela norma *Modbus RTU*, mas sim pelo *software da Fatek*. Assim sendo, temos os seguintes valores: 1-ler informação do *slave*, 2-escrever múltipla informação no *slave*, 3-escrever num único registo
- **SR+4=R4**: nº de posições a ler no *slave*
- **SR+5=R5**: tipo de dados do *master (PLC)* onde irá ser escrita a informação que venha na resposta. O nº associado a cada tipo de dados encontra-se na **página 46**.
- **SR+6=R6**: Nº do registo onde irá ser escrita a informação. Se queremos que escreva a informação no registo R100, então **R6=100**
- **SR+7=R7**: tipo de dados do *slave*. O nº correspondente encontra-se também na **página 46 do capítulo 13**.
- **SR+8=R8**: Neste registo colocamos o endereço do valor que queremos ler no *slave*.

A seguir na Tabela 6.14, está descrito o exemplo para as duas transações feitas neste teste.

Tabela 6.14: Valor em cada registo

SR	Registo R	Valor
0	0	$A550_{16}$
1	1	0702_{16}
2	2	0001_{10}
3	3	0001_{10}
4	4	0002_{10}
5	5	0012_{10}
6	6	0100_{10}
7	7	0004_{10}
8	8	19051_{10}

Quanto aos registos do tipo WR, é onde vai ser escrita a informação de como correu a comunicação. Ocupam 8 registos e a informação detalhada encontra-se na **página 46, capítulo 13 do manual.**

F.1.21 Valores a ler no aparelho Janitza

Os valores a ler no aparelho *Janitza* são os seguintes:

- *Real Energy, Supply, L1+L2+L3* (Endereço *ModBus* no aparelho: 9851)
- *Measured frequency* (Endereço *Modbus* no aparelho: 19050)

- Valor do número de impulso do botão de pressão

A função 150.M-bus do *Winproladder* retira 1 unidade aos endereços definidos, pois existem 2 conceitos na norma *Modbus*. A definição de nome, que normalmente é o valor que vem no manual do aparelho referente a uma certa posição de memória, e a definição de endereço que não é nada mais do que o valor do nome subtraído de uma unidade (diferença relativamente ao primeiro registo). Neste caso o manual do aparelho *Janitza* fornece os endereços e não os nomes. Mas como a função 150.M-bus espera que lhe sejam fornecidos os nomes, temos de somar uma unidade os valores retirados do manual do aparelho.

Assim sendo os valores que terão de ser introduzidos no programa são:

- *Real Energy, Supply, L1+L2+L3* (Endereço *Modbus* no aparelho: 9852)
- *Measured frequency* (Endereço *Modbus* no aparelho: 19051)

F.1.22 Comunicação entre camada de nível médio e alto nível por RS485 ModBus RTU *(entre autómato e o computador)*

Nesta ligação agora, o computador terá o papel de *Master*/*Cliente* e o autómato será o *Slave/Servidor*. Para isso vai ter de se configurar a porta 3 do autómato para poder realizar esta ligação. O diagrama que descreve esta ligação encontra-se na Figura 6.51.

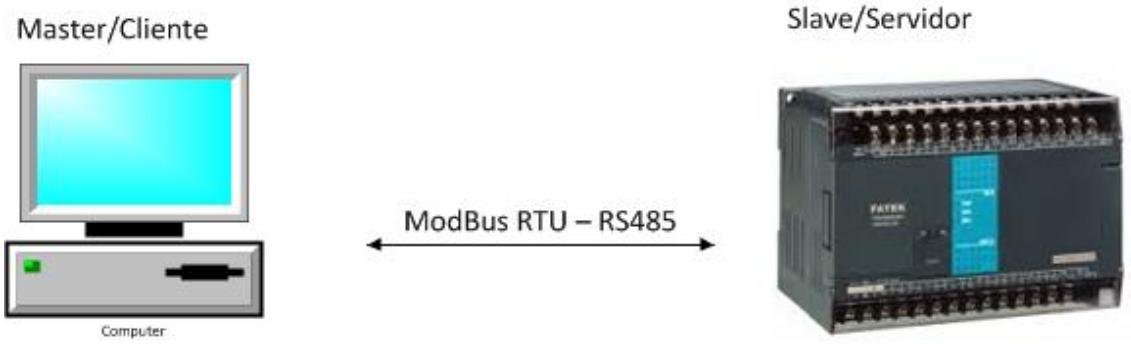


Figura 6.51: Comunicação na camada de alto nível

F.1.23 Implementação do protocolo Modbus RTU slave na interface PORT3 (RS232)

Por defeito as portas do autómato estão com o protocolo *Fatek*. Para alterar esta definição, podemos utilizar o registo R4047, como indicado na **secção 12.4.2, do capítulo 12 do manual do autómato** [30]. Para que a porta 3 utilize o protocolo *ModBus RTU slave*, um dos valores possíveis é **R4047=5620H**.

Assim sendo desta simples forma, a **porta 3** passa agora a interpretar o protocolo *ModBus RTU*, mas como sendo um *slave* da rede.

No entanto, existe um problema. À exceção das funções 5 e 6 do protocolo *ModBus RTU*, o autómato não está preparado para interpretar as restantes funções. Para isso é preciso fazer um "mapeamento da memória" como descrito e explicado na **página 50 do capítulo 13**. Consultando a tabela presente nessa página e fazendo o mapeamento por exemplo para a função 04, é evidente que é necessário editar os registos R3968, R3975, R3976, R3977.

- R3968=A55AH significa que o utilizador vai utilizar um novo mapeamento dos endereços para a comunicação com o protocolo *ModBus RTU slave*.
- R3975= valor de início do endereço *ModBus RTU* utlizado pelo master.
- R3976= valor de início do endereço referente à gama de registos do tipo R que vão ser lidos.
- R397= intervalo de registos que vão ser lidos.

Isto é, se R3975=50, R3976=70, R3977=100, então, é possível ler um intervalo de 100 registos do tipo R, que vão desde R70 a R170. O *master* para ler estes registos, pode enviar posições que variam do 50 até 150, ou seja, por exemplo, o valor 60 do master refere-se ao R80.

Assim agora o autómato responderá aos pedidos ModBus RTU por parte do computador.

F.1.24 Aplicação em Visual Basic – Implementação de ModBus RTU no computador

A aplicação Visual Basic irá ser executada no computador e terá como função comunicar com a base de dados e com o autómato. Esta aparece representada na Figura 6.52:

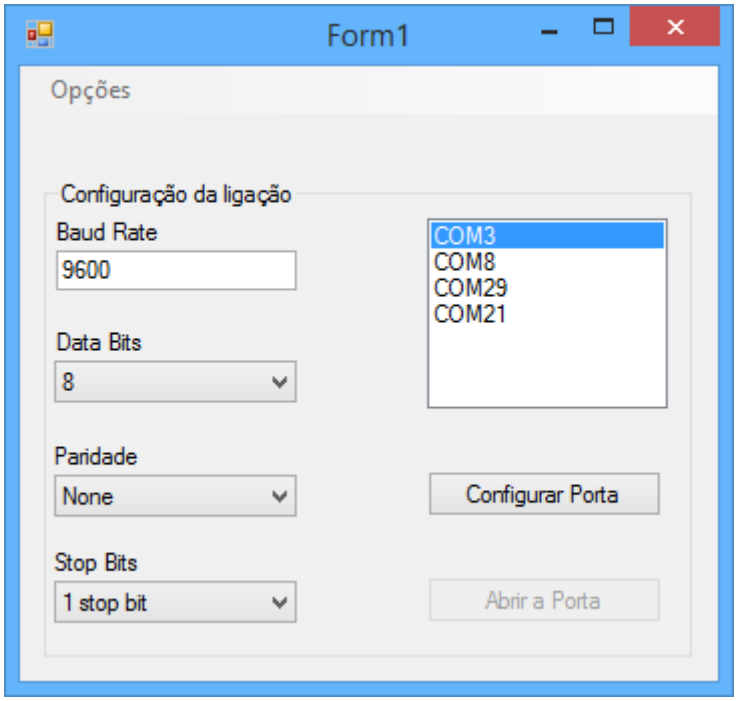


Figura 6.52: Ilustração da aplicação VB

De uma forma geral podemos ver o funcionamento simplificado desta aplicação no diagrama representado na Figura 6.53

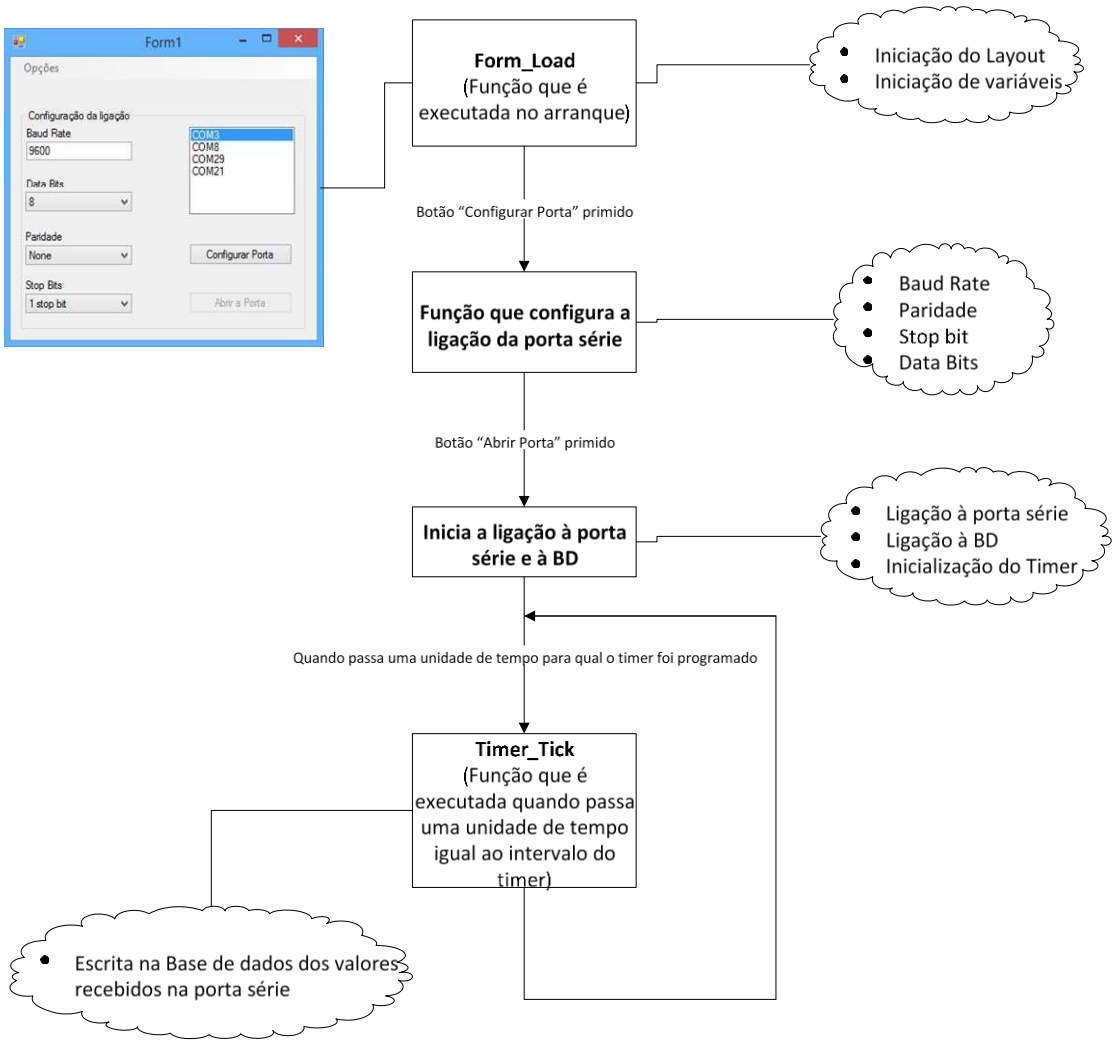


Figura 6.53: Esquema do algoritmo utilizado

Quando o executável é chamado no computador, a primeira função a ser iniciada, é a função *Form1_Load()* que é responsável para além de executar o código contido dentro desta, inicializar o *layout*. Depois de inicializada a aplicação e o seu *layout*, o utilizador pode alterar os parâmetros referentes à ligação pela porta série e clicar no botão configurar, que vai executar uma função que configurará a porta série. Se a configuração for feita com sucesso, o botão "Abrir Porta" ficará habilitado. Quando este último é clicado, a ligação à porta série e a ligação à base de dados são concretizadas. Também é iniciado um *Timer*, com um intervalo de tempo que quando é atingido executa uma função associada chamada de *Timer_Tick*. Toda a vez que esta função é executada o valor do contador volta a 0 até voltar a atingir outra vez o intervalo. Esta função tem por objetivo efetuar o pedido dos valores da contagem ao autómato e quando recebe a resposta, escreve os valores na base de dados.

Nesta secção vamos dissecar a implementação do protocolo *ModBus RTU* (necessária à comunicação computador-autómato) na aplicação *Visual Basic*.

F.1.25 ModBus RTU na aplicação

A implementação do protocolo *ModBus RTU* é de facto extremamente simples. Para isso basta apenas enviar pela porta série a informação desejada na estrutura explicada acima na

seção 0. Assim para enviar uma mensagem é necessário utilizar a função *write* associada ao objeto *SerialPort*.

Por exemplo se queremos enviar a mensagem do exemplo 0**,** (04 01 00 0A 00 0D DD 98), e se o objeto associado à porta série tem o nome *SerialPort1*, então é necessário escrever o seguinte código, por exemplo:

[redacted]

Em que a variável *palavra* é um *array* de *bytes*, e cada um destes representa dois caracteres hexadecimais. O segundo campo da função *Write* representa o índice do *array* em que deve começar a ser enviada a informação e o último campo representa o nº de *bytes* a enviar.

F.1.26 Comunicação na camada de alto nível – entre o computador e base de dados

A comunicação entre a aplicação no computador e a base de dados é feita na linguagem *SQL* (Structured Query Language), sob uma ligação do tipo *ODBC* (Open Database Connectivity).

Para este tipo de ligação, a base de dados tem de estar presente no computador local onde corre a aplicação *VB*. Antes de observar a comunicação vai ser explicado como fazer uma ligação ODBC.

F.1.27 Como fazer uma ligação ODBC a uma base de dados local

No caso em questão, todo este trabalho foi desenvolvido no sistema operativo *Windows 8.1*. no entanto os passos a seguir explicados podem ser idênticos ou até mesmo iguais nos restantes sistemas operativos da *Microsoft*.

F.1.28 Criar uma base de dados

Primeiro vai ser criada uma base de dados. Neste caso foi utilizada uma ferramenta do *Microsoft Office* para o efeito. Esta ferramenta tem o nome de *Microsoft Access 2013* como mostra a Figura 6.54. Mais abaixo vão ser analisadas as tabelas criadas. Para já a base de dados pode estar vazia.

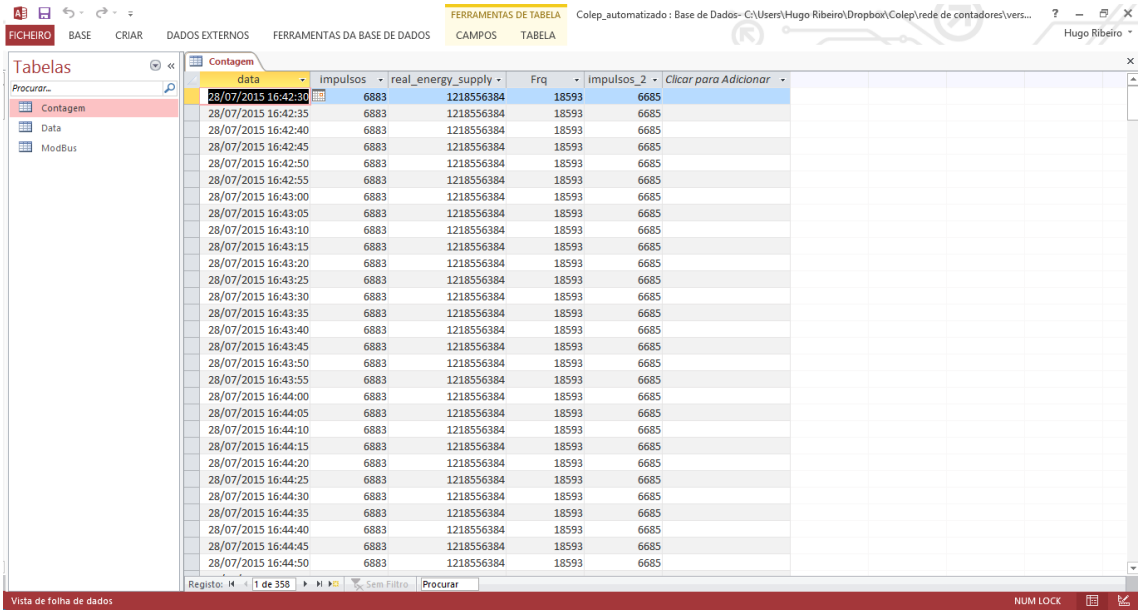


Figura 6.54: Base de Dados Access

F.1.29 Criar ligação

Depois de criado o ficheiro da base de dados podemos fazer a ligação. Para isso é necessário seguir os seguintes passos:

- Realizar uma procura recorrendo à ferramenta do *Windows* com a palavra-chave "*ODBC*" como mostra a Figura 6.55

Figura 6.55: Procura ODBC

- Selecionar a opção 32 bits, e a seguinte janela (Figura 6.56) deverá aparecer:

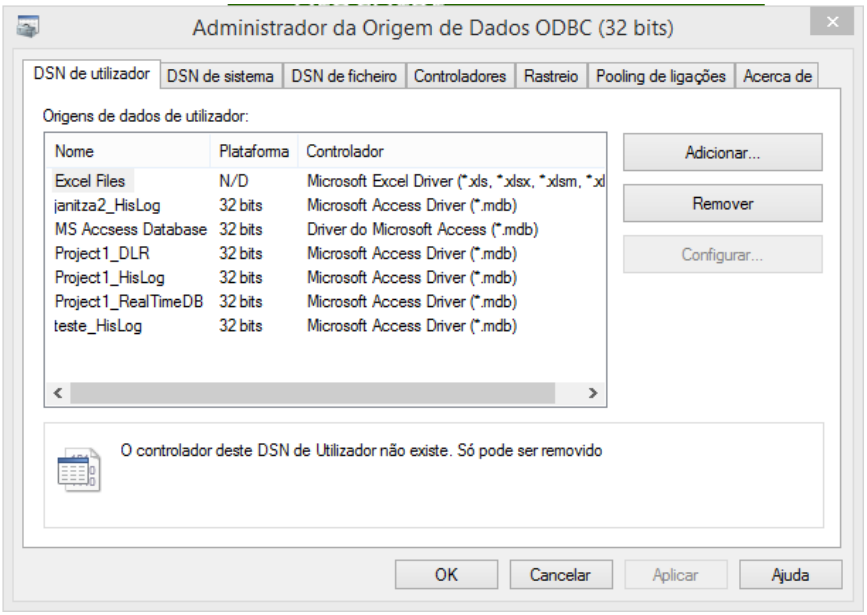


Figura 6.56: Administrador ODBC 32bit

- Selecionar o separador "*DSN de sistema*" ilustrado na Figura 6.57

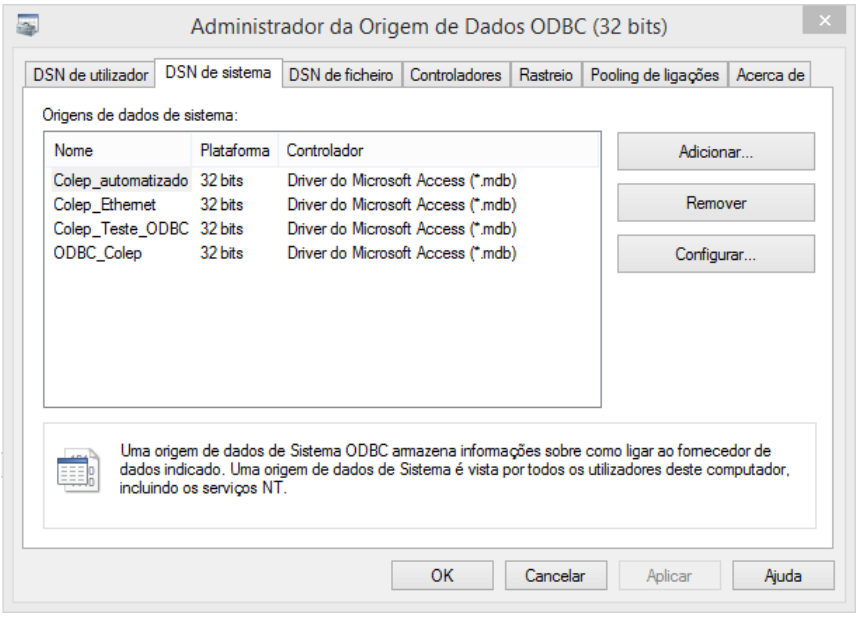


Figura 6.57: Seleção do separador "DSN de sistema"

- Carregar o botão "*Adicionar*" e selecionar a opção "*Driver do Microsoft Access (*.mdb)*" presente na Figura 6.58

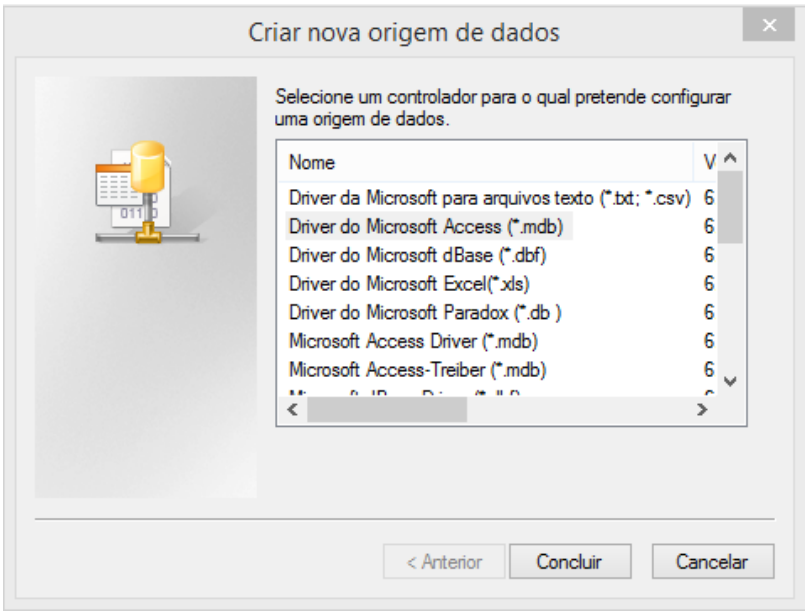


Figura 6.58: Criação da nova origem de dados

- Dar um nome à ligação ODBC e carregar no botão "*Selecionar*" como mostra a Figura 6.59

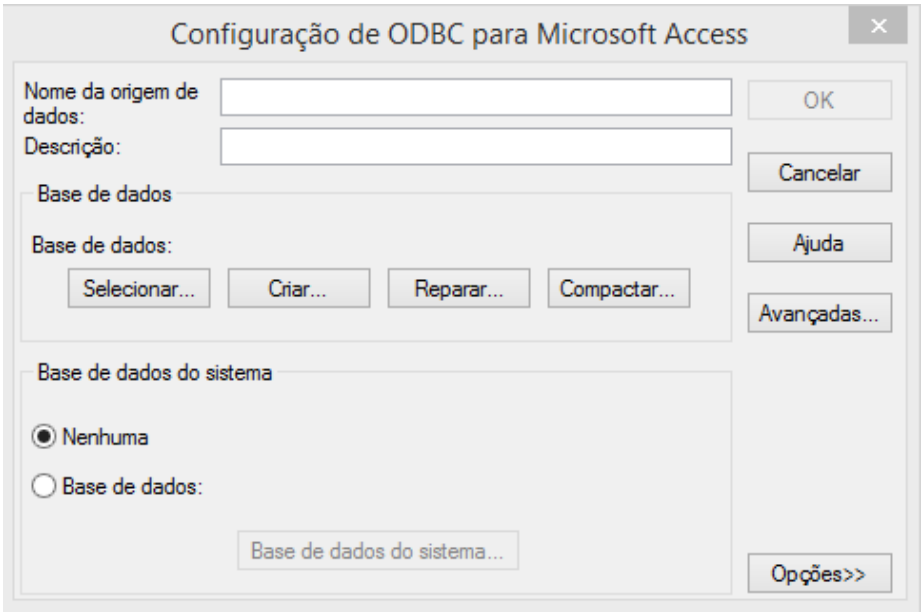


Figura 6.59: Configuração da ligação ODBC

- Utilizar o *Open Dialog Box* que surge para selecionar o ficheiro pretendido através do seu endereço no computador local, e clicar no OK, à imagem da Figura 6.60

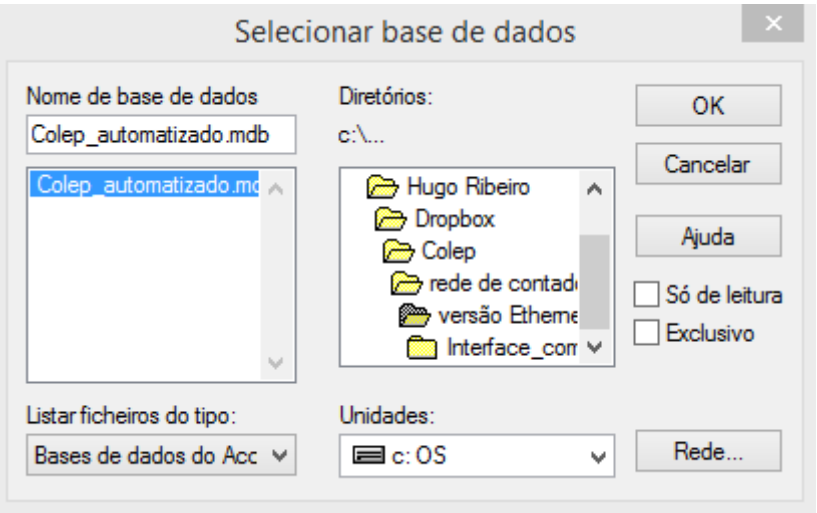


Figura 6.60: Seleção do ficheiro base de dados

- Voltar a clicar OK e já está efetuada a ligação ODBC à base de dados pretendida

F.1.30 Como utilizar as ferramentas do Visual Basic para comunicar com a base de dados ODBC

Agora que já existe a uma ligação *ODBC* a uma base dados no computador local é possível comunicar através de uma aplicação *Visual Basic*. Para que se possa utilizar os objetos do tipo *ODBC* é preciso importá-los. Para isso há que ser escrita a seguinte linha de código logo no início do programa:

[redacted]

Agora todas as funções e objetos associados à biblioteca *ODBC* poderão ser utilizados. Para a interação com a base de dados, é necessário criar os seguintes objetos:

[redacted]

Sendo *cn* o objeto responsável pela ligação, *cmd* o objeto referente aos pedidos enviados para a base de dados e *rdr* o objeto responsável pela leitura da resposta aos comandos.

Mas antes é perentório perceber o funcionamento da linguagem *SQL*.

F.1.31 Linguagem SQL (Structured Query Language)

A linguagem *SQL* surgiu na década de 70 nos laboratórios da *IBM*. Esta linguagem tem como objetivo fornecer um conjunto de ferramentas que permitam ao utilizador facilmente interagir com bases de dados relacionais. É uma linguagem bastante simples e de fácil aprendizagem o que faz com que seja a mais utilizada em bases de dados relacionais. Os comandos da linguagem são divididos em quatro grupos: *DML – DATA MANIPULATION LANGUAGE*, *DDL – DATA DEFINITION LANGUAGE, DCL – DATA CONTROL LANGUAGE* e *DTL – DATA TRANSACTION LANGUAGE*.

F.1.32 DML – DATA MANIPULATION LANGUAGE

Subconjunto que contém os comandos para operar sobre os dados das bases de dados. Os comandos SQL desse subconjunto são:

- **INSERT,** insere um registo na base de dados
- **UPDATE,** atualiza um registo na base de dados
- **DELETE,** apaga registos da base de dados
- **SELECT,** seleciona registos na base de dados

F.1.33 DDL – DATA DEFINITION LANGUAGE

Subconjunto que contém os comandos para gerir a estrutura da base de dados. Os comandos deste subconjunto são:

- **CREATE**: utilizado para criar objetos no banco de dados.
- **DROP**: utilizado para remover um objeto do banco de dados
- **ALTER**: utilizado para alterar a estrutura de um objeto

F.1.34 DCL – DATA CONTROL LANGUAGE

Subconjunto que contém os comandos úteis para permitir ou não o acesso à base de dados.

- **GRANT**: Autoriza um usuário a executar alguma operação.
- **REVOKE**: Restringe ou remove a permissão de um usuário executar alguma operação.

F.1.35 DTL – DATA TRANSACTION LANGUAGE

Subconjunto que fornece mecanismos para controlar transações na base de dados:

- **BEGIN TRANSACTION**, iniciar uma transação
- **COMMIT**, efetivar as alterações no banco de dados
- **ROLLBACK**, cancelar as alterações

F.1.36 Cláusulas da linguagem SQL [44]

Para além dos comandos, a linguagem *SQL* também dispõe de cláusulas para selecionar os dados que vão ser sujeitos ao comando. As cláusulas existentes são as seguintes [44]:

- **FROM** – especifica a tabela de origem dos dados a selecionar.
- **WHERE** – especifica as condições que devem reunir os registros que serão selecionados.
- **GROUP BY** – separa os registros selecionados em grupos.
- **HAVING** – expressa a condição que deve satisfazer cada grupo.
- **ORDER BY** – ordena os registros selecionados com uma ordem específica.
- **DISTINCT** – seleciona dados sem repetição.
- **UNION** - combina os resultados de duas consultas SQL em uma única tabela para todas as linhas correspondentes.

F.1.37 Operadores lógicos presentes na linguagem SQL

- **AND** – Verdadeiro se ambas as condições sejam corretas.
- **OR** – Verdadeiro se uma das condições é correta.
- **NOT** – Verdadeiro se a condição é incorreta e vice-versa.

F.1.38 Operadores relacionais da linguagem SQL

- **<** Menor
- **>** Maior
- **<=** Menor ou igual

- **>=** Maior ou igual
- **=** Igual
- **<>** Diferente
- **BETWEEN** – Condição de existência num intervalo entre dois valores inteiros.
- **LIKE** – Compara a totalidade ou parte de um registo com um dado valor.
- **IN** – Verifica se um valor pertence a uma dada lista.

F.1.39 Funções de Agregação

As funções de agregação, como os exemplos abaixo, são usadas dentro de uma cláusula SELECT em grupos de registros para devolver um único valor que se aplica a um grupo de registros.

- **AVG** – Calcula a média dos valores de um determinado campo.
- **COUNT** – Devolve o número de registos da seleção.
- **SUM** – Devolve a soma de todos os valores de um campo determinado.
- **MAX** – Devolve o maior valor de um campo especificado.
- **MIN** – Devolve o menor valor de um campo especificado.

F.1.40 Estrutura de um pedido SQL

Os pedidos efetuados em SQL têm uma estrutura genérica. Essa estrutura engloba normalmente um comando, uma cláusula, operador e funções de agregação se necessário e respetivos argumentos.

Para melhor compreensão, irão ser apresentados alguns exemplos agora de seguida. Dada a base de dados representada na Tabela 6.15, retirada de [13], vamos ver os resultados da base de dados ao pedidos efetuados em *SQL*.

Tabela 6.15: Exemplo SQL

ShipperID	ShipperName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

Exemplo 1: com o seguinte pedido *SQL*:

SELECT Phone FROM [Shippers] WHERE ShipperID>1

Obtemos o resultado seguinte, na Tabela 6.16:

Tabela 6.16: Resultado

Phone
(503) 555-3199
(503) 555-9931

Exemplo 2:

UPDATE Shippers SET Phone='(503) 999-9999' WHERE ShipperID=1

SELECT * FROM [Shippers]

O resultado é o seguinte, presente na Tabela 6.17:

Tabela 6.17: Resultado

ShipperID	ShipperName	Phone
1	Speedy Express	(503) 999-9999
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

Com estes dois exemplos é possível ter uma pequena noção da construção de um pedido em *SQL*.

F.1.41 Base de dados do teste e a sua comunicação com a aplicação Visual Basic

Para este teste, a base de dados em Access quem contém os valores dos contadores, possui três colunas. A coluna referente à propriedade *Real Energy, Supply,* referente à propriedade *Measured frequency,* e ao contador de impulso (botão de pressão). Quanto às linhas, o campo nelas representado é a data e hora a que foram feitas as leituras dos campos acima referidos. A tabela chamada *Contagens* está representada na Figura 6.61:

data	impulsos	real_energy_supply	Frq
21/07/2015 12:12:03	321	1218556383	18593
21/07/2015 12:12:08	323	1218556383	18593
21/07/2015 12:12:13	324	1218556383	18593
21/07/2015 12:12:18	326	1218556383	18593
21/07/2015 12:12:23	334	1218556383	18593
21/07/2015 12:12:28	338	1218556383	18593
21/07/2015 12:12:33	338	1218556383	18593
21/07/2015 12:12:38	339	1218556383	18593
21/07/2015 12:12:43	339	1218556383	18593
21/07/2015 12:12:48	341	1218556383	18593
21/07/2015 12:22:12	344	1218556383	18593
21/07/2015 12:22:17	344	1218556383	18593
21/07/2015 12:22:22	348	1218556383	18593
21/07/2015 12:22:27	354	1218556383	18593
21/07/2015 12:22:32	361	1218556383	18593
21/07/2015 12:22:37	367	1218556383	18593
21/07/2015 12:22:42	374	1218556383	18593
21/07/2015 12:22:47	381	1218556383	18593
21/07/2015 12:22:52	386	1218556383	18593
21/07/2015 12:22:57	396	1218556383	18593

Figura 6.61: Dados da rede de teste

Outra tabela presente na base de dados é a tabela *ModBus*. Nesta tabela encontram-se os valores que devem estar a partir do registo ***R3000*** do autómato***.*** Este registo está presente no parâmetro ***SR*** da função ***150P_M.Bus*** como explicado na secção 0 e este, em conjunto com os registos seguintes, são responsáveis por configurar as palavras *ModBus* a ser enviadas pelo autómato. Ora no programa do autómato apenas é inicializada a posição ***R3000*** com o valor ***A550H*** que é um valor predefinido para o parâmetro ***SR.*** Todos os outros registos seguintes que configuram as mensagens *ModBus* são escritos no autómato recorrendo à porta que possui o protocolo *ModBus RTU Slave* e que está ligada ao computador. Desta forma evitamos a reprogramação do autómato cada vez que queiramos alterar as mensagens *ModBus* enviadas por este, pois a própria aplicação *Visual Basic* escreve nessas posições de memória que estão presentes na base de dados. Assim o sistema fica independente do programa do autómato e apenas dependente da base de dados e da aplicação *Visual Basic*.

Por cada linha desta tabela, temos a configuração de um pedido *ModBus RTU*. Cada uma das colunas corresponde à configuração de uma das posições de memória posteriores ao registo ***SR*** referente à função ***150P_M.Bus*** A figura Figura 6.62 mostra isso.

ID	contador_grandeza	R3xx2	R3xx3	R3xx4	R3xx5	R3xx6	R3xx7	R3xx8	*Clicar para Adicionar*
0	real_energy_supply	1	1	2	12	1012	4	9852	
1	Frq	1	1	1	12	1010	4	111	
0		0	0	0	0	0	0	0	

Figura 6.62Tabela de Configuração